

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITE DE COOPÉRATION
EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété
Intellectuelle
Bureau international



(43) Date de la publication internationale
6 novembre 2003 (06.11.2003)

PCT

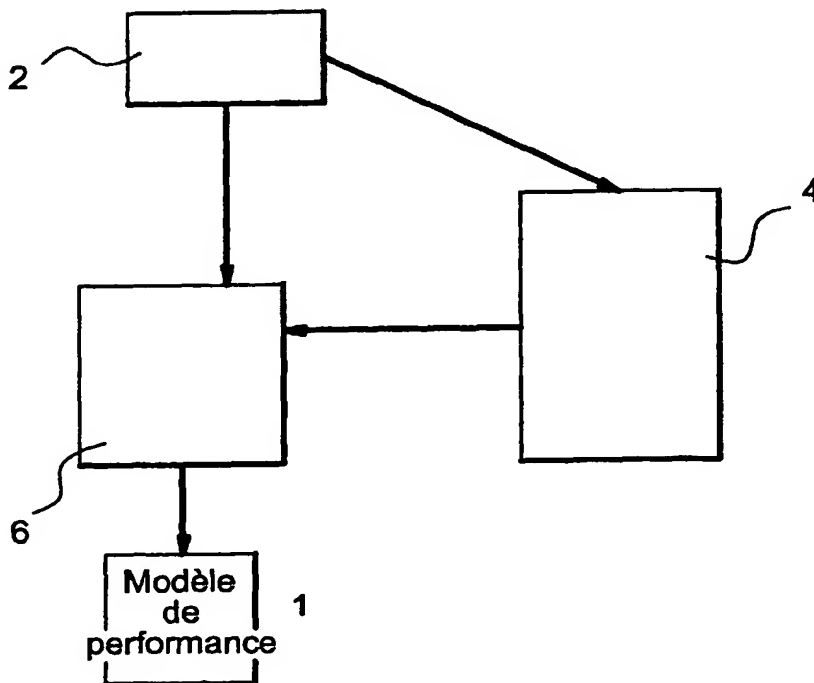
(10) Numéro de publication internationale
WO 03/092245 A2

- (51) Classification internationale des brevets⁷ : H04M (72) Inventeurs; et
(21) Numéro de la demande internationale : PCT/FR03/01251 (75) Inventeurs/Déposants (pour US seulement) : MONIN, Wei [FR/FR]; 46 avenue de Lorraine, F-22300 LANNION (FR). DUBOIS, Fabrice [FR/FR]; 155 rue St Guirec, F-22700 PERROS-GUIREC (FR). VINCENT, Daniel [FR/FR]; 15, route de Kerguntuil, F-22560 PLeumeur Bodou (FR). COMBES, Pierre [FR/FR]; 28 rue des Fraisières, F-91120 Palaiseau (FR).
(22) Date de dépôt international : 18 avril 2003 (18.04.2003)
(25) Langue de dépôt : français
(26) Langue de publication : français
(30) Données relatives à la priorité : 02/05063 23 avril 2002 (23.04.2002) FR (74) Mandataire : DU BOISBAUDRY, Dominique; c/o BREVALEX, 3, rue du Docteur Lancereaux, F-75008 PARIS (FR).
(71) Déposant (pour tous les États désignés sauf US) : FRANCE TELECOM [FR/FR]; 6 Place d'Alleray, F-75015 PARIS (FR). (81) États désignés (national) : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ,

[Suite sur la page suivante]

(54) Title: METHOD OF GENERATING A PERFORMANCE MODEL FROM A FUNCTIONAL MODEL

(54) Titre : PROCEDE DE GENERATION D'UN MODELE DE PERFORMANCE A PARTIR D'UN MODELE FONCTIONNEL.



(57) Abstract: The invention relates to a method of generating a performance model from a functional model of a system comprising a plurality of co-operating and distributed hardware and software entities in order to provide a service to at least one user. The inventive method comprises the following steps consisting in: dividing the requests that are representative of the system into a finite number of groups and, for each group of requests, identifying the corresponding execution stream; formalising the execution streams using a notation which can be used to demonstrate (i) the causal relations between the different system software entities involved in the execution stream and (ii) the data that characterise the resource consumption of the system; producing an intermediate model; and automating the transformation of the intermediate model produced into a performance model.

1...PERFORMANCE MODEL

[Suite sur la page suivante]



WO 03/092245 A2



DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) États désignés (régional) : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasién (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK,

TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— sans rapport de recherche internationale, sera republiée dès réception de ce rapport

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) **Abrége :** L'invention concerne un procédé de génération d'un modèle de performance à partir d'un modèle fonctionnel d'un système comportant une pluralité d'entités matérielles et logiciels réparties coopérant pour fournir un service à au moins un utilisateur. Le procédé selon l'invention comporte les étapes suivantes : - Répartir les requêtes représentatives du système en un nombre fini de groupes et identifier, pour chaque groupe de requêtes, le flot d'exécution correspondant, - Formaliser les flots d'exécutions à l'aide d'une notation permettant de mettre en évidence, d'une part, les relations causales entre les différentes entités logicielles du système impliquées dans les flots d'exécution, et d'autre part, les informations caractérisant la consommation des ressources du système, - Elaborer un modèle intermédiaire, - Automatiser la transformation du modèle intermédiaire élaboré en un modèle de performance.

PROCEDE DE GENERATION D'UN MODELE DE PERFORMANCE A
PARTIR D'UN MODELE FONCTIONNEL

DESCRIPTION

5 Domaine technique

 L'invention se situe dans le domaine de l'étude des performances par simulation d'un système, tel que par exemple un système de télécommunication comportant une pluralité de composants matériels et logiciels
10 répartis et qui coopèrent pour fournir un service à un ou plusieurs utilisateurs.

 L'invention concerne plus spécifiquement un procédé de génération d'un modèle de performance à partir d'un modèle fonctionnel d'un tel système.

15

Etat de la technique antérieure

 Lors de la réalisation puis du déploiement d'un système, il faut s'assurer qu'il possède les capacités fonctionnelles attendues, mais aussi qu'il tiendra la
20 charge dans les conditions d'exploitation. Si les performances chutent, il peut même arriver que le système ne remplisse plus les services prévus. Une étude de performance a pour premier objectif de prévoir les phénomènes d'écroulement du système afin d'éviter
25 tout dysfonctionnement.

 L'étude de performance permet de connaître d'une façon précise la répartition des ressources du système et d'identifier ainsi les éventuels goulots d'étranglement. Un second objectif est d'ordre
30 économique. Il s'agit de déterminer les configurations optimales des ressources, afin de maximiser leur

efficacité et ainsi d'éviter l'achat d'équipements supplémentaires. Le dimensionnement des ressources consiste à déterminer les paramètres quantitatifs nécessaires pour la qualité de service désirée tels que
5 par exemple la taille des tampons, le nombre de serveurs, la répartition des entités de traitement sur les différentes machines, etc.

Une étude de performance s'effectue soit sur un modèle, soit sur le système réel par des mesures.
10 Lorsque c'est sur un modèle, on peut généralement observer deux types de comportement du système étudié, le comportement transitoire et le comportement stationnaire. Le premier porte sur des périodes courtes et le second sur des périodes longues. Les approches
15 les plus sollicitées sont notamment les techniques mathématiques, la simulation discrète, et l'analyse basée sur des graphes.

Les notations pour exprimer les modèles sont souvent des types suivants :

- 20 - systèmes à files d'attente,
- réseaux de Petri temporisés,
- automates temporisés,
2 - modèles hiérarchiques ou descriptions textuelles dans des langages spécialisés.

25 L'évaluation des performances dépend de la technique utilisée qui doit être choisie en tenant compte des caractéristiques de l'application considérée. Les techniques de résolution sont classées en trois catégories principales :

- 30 - la méthode analytique,
- les tests réels, et

- la simulation.

La méthode analytique fait appel le plus souvent à la théorie des systèmes à files d'attente, ou aux réseaux de Petri stochastiques.

5 Les systèmes à files d'attente restent l'un des plus puissants formalismes mathématiques permettant d'analyser quantitativement une très grande variété de systèmes. Par exemple, un système informatique peut être considéré, de manière très abstraite, comme un
10 ensemble de ressources matérielles et logicielles (serveurs) utilisées par des tâches ou des programmes (clients). Les ressources étant en nombre limité, les programmes vont entrer en concurrence pour l'accès à ces ressources, et l'on résout cette concurrence par
15 des files d'attente.

Le formalisme des réseaux de Petri repose sur une représentation graphique qui permet de bien comprendre le comportement du système. Ce formalisme convient notamment aux systèmes dont le comportement
20 est fortement caractérisé par les aspects de concurrence, de synchronisation, de communication et de coopération. Le système peut être analysé qualitativement (absence d'inter-blocage, sûreté de fonctionnement, etc.).

25 Un handicap majeur du réseau de Petri réside dans l'explosion du nombre d'états à examiner lorsque la complexité du système augmente.

L'approche analytique, enfin, consiste généralement à exprimer le comportement du système sous
30 forme d'un modèle mathématique exact aboutissant à des formules closes. Le comportement du système est alors

réduit à un ensemble de variables liées entre elles par des équations. L'atout principal de cette approche analytique réside dans le fait qu'elle ne nécessite pas d'outil de support sophistiqué lors des phases de modélisation et de résolution du problème, ce qui convient particulièrement aux phases amont de la conception d'une application. Avec cette approche, il suffit d'appliquer la formule pour déterminer les quantités recherchées et surtout pour connaître l'influence relative des différents paramètres. Cette approche, qui requiert couramment de fortes hypothèses (indépendance des événements, conservativité des serveurs, etc.), s'avère praticable pour des systèmes à l'architecture fonctionnelle relativement simple tels que par exemple l'étude des systèmes de commutation des réseaux de transport. En outre, la méthode analytique s'avère bien adaptée à l'étude des «pires cas» de comportement du système, en faisant des approximations simplificatrices et pessimistes.

Cependant, les systèmes logiciels se prêtent beaucoup moins bien aux méthodes analytiques, car ils présentent beaucoup de variantes de comportement. En effet, le comportement du système d'exploitation dépend fortement de l'architecture de la machine, sur laquelle on ne dispose souvent que de très peu d'informations. Aussi, les interactions entre les différents processus peuvent être très complexes et difficiles à caractériser par des lois probabilistes.

Dans les systèmes répartis, des complications supplémentaires apparaissent du fait de la coopération entre des systèmes hétérogènes où seule une vue

partielle de l'état du système est disponible au niveau de chaque composant. Dans le cas des services de télécommunication, il est également difficile de caractériser le comportement des clients qui génèrent le trafic des requêtes.

A l'opposé de la méthode analytique, l'approche par tests réels consiste, moyennant une infrastructure d'observation (sondes), à faire des mesures directement sur le système réel. L'observation se fait généralement soit au niveau logiciel, soit au niveau du système exploitation, soit aux deux niveaux en même temps. Cette approche a pour avantage l'obtention de résultats réels, notamment pour la capacité de traitement limite du système (montée en charge).

La mesure des performances sur un système réel nécessite de disposer d'une implantation complète du système à tester (matériels et logiciels), par conséquent l'étude ne peut être menée qu'a posteriori, et peut impliquer des achats de matériels coûteux. La réalisation d'une sonde pour inspecter le système est une tâche difficile car l'instrumentation ne doit pas perturber de manière significative ce que l'on mesure. En particulier, La répartition des machines nécessite une synchronisation d'horloges satisfaisante. De plus, toute modification logicielle ou matérielle entraîne un surcoût de mise en oeuvre en termes d'interopérabilité et de portabilité, de reconfiguration, et de réglages. En outre, les programmes de tests doivent être développés et eux-mêmes testés et sont fortement liés au code réel de l'application ce qui est d'autant plus contraignant que les processus de développement et de

mise en exploitation d'une application sont courts. Il faut donc souvent développer en parallèle l'application et les programmes de tests, ce qui induit de fortes contraintes de synchronisation au niveau de l'organisation du développement. Enfin, les tests réels souffrent d'une limitation qui provient du fait que seules des configurations simples peuvent être considérées. Par exemple, la capacité de traitement est évaluée en testant le comportement du système lorsque des demandes de service arrivent simultanément. Par contre, pour connaître l'évolution du temps de réponse (temps maximum, temps moyen, répartition...), de l'occupation des tampons et du taux de pertes, il faut générer un très grand nombre de requêtes pour que les résultats statistiques des tests soient représentatifs. D'autre part, il faut aussi générer des flux d'arrivée de ces requêtes selon des scénarii représentatifs du comportement des utilisateurs. Ces flux sont caractérisés notamment par le taux d'arrivée moyen et le rythme (déterministe, en rafale, aléatoire sur une durée, exponentiel...). Ces contraintes imposent de disposer de plusieurs machines clientes, car le nombre de processus actifs par CPU est souvent limité à quelques centaines. Il faut également pouvoir les synchroniser pour que le comportement des utilisateurs généré soit proche de la réalité, ce qui est très difficile à mettre en œuvre.

L'approche par simulation semble de loin la plus utilisée pour l'évaluation des performances de toutes sortes de systèmes et consiste à modéliser le comportement du système, le plus souvent à l'aide d'un

5 système à files d'attente ou d'un réseau de Petri, puis
à expérimenter le modèle en faisant varier un ensemble
de paramètres correspondant aux différentes
configurations du système. Par exemple, pour des
10 systèmes logiciels, les paramètres peuvent être la
vitesse de calcul des processeurs, le débit de
transmission des réseaux, la politique d'ordonnancement
des processus, le nombre et la topologie des machines,
le taux et le rythme d'arrivée des requêtes, etc.. La
15 simulation est plus aisée à appréhender, et plus
puissante que la méthode analytique. En effet, la
formulation du modèle à simuler ne s'exprime pas sous
forme d'équations mais de programmes, ce qui permet de
construire un modèle aussi proche que possible du
20 système réel étudié. Elle est beaucoup moins coûteuse
en matériel et en logiciel que l'approche par tests car
elle exploite un modèle (une abstraction, une
représentation virtuelle) au lieu d'un système réel.
Cette approche fournit un outil puissant pour la
25 prédiction des performances depuis la conception du
système à étudier jusqu'à sa réalisation.

Des simulateurs logiciels généralistes sont
apparus depuis 1980 et on peut citer par exemple QNAP
[<http://www.hyperformix.com>], SIMSCRIPT
25 [http://www.caci.com/index_main.shtml], OPNET
[<http://www.opnet.com>], et SES/Workench
[<http://www.hyperformix.com>].

Contrairement aux campagnes de tests, le temps
d'exécutions d'une simulation n'est pas lié aux temps
30 réels des traitements de l'application. Ainsi,
l'approche par simulation permet de réaliser un plus

grand nombre de scénarii en faisant simplement varier les paramètres du modèle. La mise au point d'un modèle permet souvent d'énoncer des conjectures sur le système considéré et de pressentir plus facilement certains
5 éléments de réponses au problème de la performance. Les simulations permettront de confirmer ou d'infirmer ces conjectures. D'autres éléments, tels que la distribution du temps de réponse de bout en bout ou du
10 taux de perte, sont le plus souvent très difficiles à quantifier et seule la simulation permet de les mettre en évidence.

L'approche par simulation se déroule en quatre phases:

- l'élaboration du modèle de comportement du
15 système : elle commence par la sélection des entités du système (un composant logiciel, un processus, etc.) qu'il est judicieux de modéliser, par rapport aux objectifs précis de l'étude de performance. L'élaboration se poursuit par la description du
20 fonctionnement de ces entités, ainsi que de leurs interactions. Les entités sont choisies en fonction de la finesse désirée des résultats de la simulation, mais aussi en fonction des données initiales qui peuvent
être procurées. Les paramètres sont des
25 caractéristiques significatives des éléments du système (par exemple, temps de traitement unitaire d'une entité sur un processeur, nombre de serveurs, etc.), qui sont susceptibles d'influencer les résultats de simulation. Les modèles sont représentés généralement soit par des
30 systèmes à files d'attente (QNAP, SIMSCRIPT, SES/Workbench), soit par des automates (OPNET), soit

par des réseaux de Petri.

• l'acquisition des données quantitatives pour alimenter les simulations : pour des systèmes logiciels, ces données correspondent notamment aux
5 temps de traitement des programmes sur les processeurs et aux délais de transmission si des acheminements à travers des réseaux ont lieu. Elles sont mesurées par des tests unitaires. Un test unitaire de traitement, ou de délai de transmission, correspond respectivement au
10 temps d'utilisation du processeur (unité centrale), ou du réseau quand la ressource est entièrement dédiée au processeur.

• les simulations et la collecte des résultats statistiques : on observe le comportement du système en
15 exécutant le modèle à l'aide d'un simulateur et en faisant varier ses paramètres ; on analyse les résultats de chaque simulation et on en déduit des résultats de performance pour les configurations choisies.

• validation du modèle de simulation : la confiance que l'on peut accorder aux résultats de simulations nécessite une validation du modèle. Elle consiste à
20 réaliser quelques tests réels représentatifs soigneusement choisis et à confronter ces résultats réels avec ceux obtenus par simulation.

De manière générale, l'approche par simulation constitue un bon compromis entre coût de mise en oeuvre et valeur des résultats. En faisant varier divers paramètres, elle permet d'étudier le comportement d'un
30 système ou d'une application sous une grande variété de configurations. Ceci sans avoir toute l'infrastructure

cible du système (matériel et logiciel) et, pour les applications; sans en faire le développement complet. Par ailleurs, elle convient parfaitement lorsque l'on veut comparer des différentes technologies ou produits, ainsi que pour dimensionner et optimiser le système en cours d'élaboration.

Les principales difficultés de l'approche par simulation résident dans la capture du comportement du système à étudier, l'obtention des mesures quantitatives unitaires et la connaissance statistique du comportement des clients du système. En effet, tout modèle de simulation est une approximation du système réel, et la fidélité du modèle à la réalité dépend fortement de la finesse des connaissances disponibles sur le fonctionnement du système. Or les systèmes logiciels sont souvent des produits qui se présentent sous forme de « boîtes noires », c'est-à-dire de programmes exécutables dont on ne dispose pas des spécifications. L'introduction d'outils de traces pour comprendre les mécanismes et les comportements internes et pour disposer des mesures unitaires sur tous les éléments qui composent le système dans la modélisation est une tâche indispensable et délicate. En outre, la connaissance du comportement des utilisateurs joue également un rôle primordial pour alimenter les simulations, loi d'arrivée des requêtes, répartition des scénarii utilisateur, etc. En effet, ces éléments ont une incidence non négligeable sur la performance du système, notamment en ce qui concerne les temps de réponse. Cette connaissance ne peut être fournie que par des mesures sur un système réel. Or, en général

très peu de statistiques sont disponibles à ce sujet. De plus, pour de nouvelles applications, une modélisation anticipée des profils des utilisateurs est nécessaire.

5 Le passage d'un modèle basé sur les aspects fonctionnels d'un système à un modèle de performance a fait l'objet de plusieurs travaux cités à titre d'exemples ci-après.

 Timed SDL (TSDL) [F. Bause and P. Buchholz,
10 "Qualitative and Quantitative Analysis of Timed SDL Specifications", in N. Gerner, H. G. Hegering, J. Savolvod, (Eds.) Springer-Verlag, 1993] implémente une forme de systèmes de transitions temporisés, construits à partir d'une spécification SDL dans laquelle à chaque
15 transition, est attachée une durée exponentielle. Il n'inclut pas de notions de ressources ou de charge. A partir d'une représentation intermédiaire sous la forme d'automate, l'outil effectue des analyses qualitatives et quantitatives en utilisant des algorithmes de type
20 Markovien.

 DL-net [H. M. Kabutz, "Analytical performance evaluation of concurrent communicating systems using
SDL and stochastic Petri nets", Doctoral Thesis, department of Computer science, University of Cape
25 Town, Republic Of south Africa, 1997] transforme un modèle SDL appauvri (soumis à de fortes restrictions sur les types de données) en un modèle QPN (Queuing Petri Nets), c'est-à-dire un réseau de Petri muni de files d'attente. C'est seulement à ce second niveau que
30 sont introduites les informations stochastiques de durée. Un outil convertit ce modèle en une chaîne de

Markov qui est ensuite résolue. La sémantique de la description SDL originale peut être différente de celle de SDLnet, mais des solutions pragmatiques acceptables ont été proposées, du point de vue des auteurs de cette
5 approche.

SPECS (SDL Performance Evaluation of Concurrent Systems) [Butow M., Mestern M., Schapiro C., Kritzing P. S., «Performance Modeling from Formal Specifications», FORTE-PST y' 96, Kaiserslautern, Germany, October
10 1996] permet de modéliser les ressources (les machines) par des blocs SDL et les tâches s'exécutant sur une machine par des processus à l'intérieur d'un même bloc. Les processus dans des blocs différents s'exécutent de manière concurrente, alors que les processus du même
15 bloc s'exécutent dans un mode multitâche. Des délais de canaux et des caractéristiques de fiabilité peuvent être ajoutés. Le modèle s'exécute sur une machine virtuelle qui est dérivée du modèle SDL.

SPEET (SDL Performance Evaluation Tool) [M. Steppler, M. Lott, "SPEET - SDL Performance Evaluation Tool", in A. Cavalli, A. Sarma (Ed.), SDL'97 - Time for Testing, Proceeding of the 8th SDL Forum, Elsevier,
20 1997.] a pour objectif principal l'analyse de performances de systèmes spécifiés en SDL s'exécutant dans un environnement temps-réel. Les systèmes peuvent
25 être simulés ou exécutés sur des émulateurs de matériels existants (Intel®, Motorola® et Siemens®). Ils sont stimulés par des générateurs de trafics et sont interconnectés par des liens de transmission
30 correspondant à des canaux physiques. Les utilisateurs peuvent définir facilement des sondes à l'intérieur de

la description formelle et introduire des indications de charge.

Les approches de [E. Heck, "Performance Evaluation of Formally Specified Systems - the
5 integration of SDL with HIT", doctoral Thesis, University of Dortmund, Krehl Verlag, 1996.] et de [Martins J., «A system Engineering Methodology Integrating performance evaluation and Formal Specification », PhD Thesis, l'Ecole polytechnique
10 fédérale de Lausanne, Avril 1996] introduisent un cadre conceptuel (HIT) visant une synthèse des techniques de description formelle et d'évaluation de performance où les propriétés issues de ces deux mondes seraient préservées. Un outil transforme un modèle SDL en un
15 modèle HIT possédant une structure hiérarchique pertinente pour l'étude de performances. Une traduction (manuelle pour l'instant) a également été proposée vers un modèle OPNET, ce qui permet de bénéficier de simulateurs performants.

20 QUEST [M. Diefenbruch, "Functional and Quantitative Verification of Time-and resource Extended SDL Systems with Model-checking ", in : K Irmscher, (Ed.), Proceeding of Messung, Modellierung und Bewertung van Rechen-und Kommunikationssystemen,
25 Freiberg, Germany, VDE-Verlag, 1997] est basé sur un langage, QSDL (Queueing SDL), qui est une extension du langage SDL. En ajoutant des annotations indiquant les machines qui fournissent des services, des disciplines de service, la gestion des files d'attente dans le
30 modèle SDL, le modèle QSDL permet d'évaluer par simulation la performance du système correspondant

spécifié en SDL. Il permet également de valider et de vérifier un système SDL temporisé par model checking.

DO-IT Toolbox [A. Mitschele-Thiel and B. Mulier-Clostermann, «Performance Engineering of SDIJMSC systems», Journal on Computer Networks and ISDN Systems, Elsevier, 1998] effectue l'évaluation de performance de systèmes spécifiés en MSC et en SDL. Son originalité est de partir des MSC qui sont disponibles très tôt dans le cycle de vie, en y ajoutant des annotations de charge et précisant les ressources disponibles pour des exécutions spécifiques du système. Il fournit une technique simple d'évaluation de performance pour l'analyse de goulots d'étranglement et l'optimisation d'implémentations, à partir d'hypothèses de temps de service déterministes.

EaSy-Sim [C. Schaffer and R. Raschhofer, A. Simma, "EaSy-Sim A Tool Environment for the design of Complex, Real-Time systems", Proceeding International Conference on computer Aided Systemns technologies, Innsbruck, Springer-Verlag, 1995] est un couplage entre l'environnement GEODE pour SDL et le simulateur SES/Workbench. L'environnement SDL est utilisé pour la vérification et la validation fonctionnelle, alors que SES/Workbench modélise la partie non fonctionnelle pour évaluer la performance. Le couplage est implémenté par des échanges de messages entre le code exécutable de l'application obtenu par GEODE et le simulateur SES/Workbench. Cet outil peut s'appliquer à une spécification MSC dans lequel des exigences de temps de réponse sont ajoutées pour des systèmes temps réel [C. Schaffer "MSC/RT: A Real-Time extension to Message

Sequence Charts (MSC), Internal Report TR 140-96, Institut für Systemwissenschaften, Johannes Kepler universität Linz, 1996].

EDT (Estelle Development Tool) [M. Hendaz, «
5 Annotation Dynamique pour Evaluation de Performance des
systèmes spécifiés en Estelle », thèse doctorat à
l'Université Paris 6, 1996.] est un ensemble d'outils
basés sur le langage Estelle. Le traducteur transforme
un modèle Estelle en un modèle intermédiaire, annoté de
10 façon à assigner des temps d'exécutions non nuls aux
transitions. Les temps peuvent avoir différentes
distributions. Le simulateur/débogueur (Edb), basé sur
le modèle sémantique défini dans la norme Estelle,
comporte un jeu de fonctions spéciales permettant des
15 simulations interactives et aléatoires dans le cadre de
l'évaluation de performance.

Configuration Planner [H. M. EI-Sayed, D.
Cameron and C M. Woodside, "Automated performance
modeling from scenarios and SDL design of distributed
20 systems ", proceeding of International Symposium on
Software Engineering for Parallel and Distributed
Systems (PDSE'98), Kyoto, April 1998] est une approche
visant, pour des systèmes temps réel « temps mou » et «
temps dur », à transformer de manière automatique un
25 modèle fonctionnel spécifié en MSC en un modèle de
performance LQN (LayeredQueueing Network). La
simulation des modèles LQN permet d'optimiser, selon le
critère du temps de réponse, la répartition et
l'ordonnancement des tâches sur l'ensemble des
30 ressources (les machines) du système. Dans ce modèle la
contrainte de mémoire est ignorée.

SPEED (Software Performance Engineering Early Design) [C. U. Smith and L. G. Williamns, "Performance Engineering Evaluation of ObjectOriented Systems with SPE-ED", Computer Performance Evaluation Modelling Techniques and Tools, no. 1245, Springer-Verlag, Berlin, 1997] est un outil permettant d'évaluer l'architecture et des solutions alternatives pour la conception des systèmes à objets. La spécification fonctionnelle du système est sous forme des diagrammes de séquences représentant des traces d'exécutions (scénarios). L'outil transforme le modèle de spécification en un modèle de performance représenté par un système à files d'attente. Une combinaison d'analyse et de simulation permet d'identifier des problèmes de performance potentiels, comme les phénomènes de goulot d'étranglement.

TLC (Trace-Based Load Characterization) est une approche travaillant sur des traces d'exécutions du système, qui servent de base à la construction d'un modèle de performance en couches (LQN, ou Layered Queuing Network). Les traces - Message Sequence Charts - sont obtenues par simulation d'un modèle SDL, mais doivent être annotées afin d'explicitier les dépendances causales qui existent entre les différentes actions réalisées. Par une analyse des événements détaillés de la simulation, les annotations des traces sont produites selon un principe de marquage incrémental, qui s'apparente à la technique de coloration du sang utilisée en radiographie, l'angiogramme. Une fois ces chaînages causaux établis, des schémas de communication, classiques dans les systèmes logiciels,

sont identifiés, en pratique, chaque message se voit attribuer un type, qui peut être « RPC » (appel de procédure distante), « Forward » (RPC propagé à un tiers), « Reply » (réponse à un RPC), ou « Async » (émission asynchrone d'un message). Sur la base de cette classification, un modèle LQN est produit automatiquement, paramétré puis analysé sur le plan des performances. Le modèle LQN est une spécialisation du modèle classique à files d'attente, adapté à la description d'architectures logicielles traditionnelles, telles que les systèmes client-serveur.

Inconvénient des techniques antérieures

Les approches de l'art antérieur citées ci-dessus consistent soit à effectuer des analyses directement sur un modèle, soit à simuler un modèle avec un simulateur spécifique.

Dans le premier cas, les hypothèses de Markov sont omniprésentes. Elles peuvent éventuellement être vérifiées sur des systèmes simples. Cependant, elles ne s'appliquent généralement plus dans le cas de systèmes complexes à l'intérieur desquels de nombreuses interactions ont lieu. Faire l'hypothèse de Markov revient alors à considérer une approximation assez grossière du système réel, faute de mieux. Quant au second cas, la plupart des approches introduisent des annotations temporelles à l'intérieur du modèle fonctionnel. Le modèle de performance qui en résulte comporte généralement beaucoup trop de détails d'ordre fonctionnel, sans pertinence du point de vue des

performances, qui obèrent l'efficacité des programmes de simulation. Il faut bien garder à l'esprit qu'un modèle fonctionnel et un modèle de performance ont des objectifs fondamentalement différents : du point de vue performance, on suppose que le système fonctionne correctement et par conséquent, les détails de fonctionnement (l'ensemble des propriétés logiques) n'importent pas, seules comptent la durée pendant laquelle une tâche détient le CPU et la politique d'ordonnancement qui gère les tâches simultanées. De même, on ne considère pas les cas marginaux, qui se produisent rarement et dont l'influence sera négligeable statistiquement. Or, ces cas peuvent être nombreux et leur modèle fonctionnel peut être très complexe.

La dernière approche TLC séduit par le fait qu'elle considère un ensemble fini de scénarios, plutôt qu'une description complète du système. Toutefois, le modèle de files d'attente en couches qui est visé (LQN) introduit des concepts de haut niveau qui, dans le but de favoriser la compréhension du système par l'humain, s'avèrent à la fois rigides et artificiels vis-à-vis de l'analyse des performances, et restrictifs en termes d'expressivité.

Enfin, l'approche TLC cherche à automatiser l'identification des liens de causalité entre les différents événements d'un scénario, ce qui nous semble très délicat dans le cas général. En effet, la réaction d'un système à un stimulus, bien souvent, n'est pas conditionnée que par ce stimulus, mais également par l'état interne du processus ; or, l'approche TLC

s'attache à cerner essentiellement les facteurs déclenchants de type stimulus externe (messages). En fait, la difficulté consiste à exprimer sur les scénarios toute l'information de causalité nécessaire à la simulation de performance, tout en parvenant à faire abstraction des détails fonctionnels du système³. Comme nous le décrivons dans la suite, notre approche face à ce problème est plus pragmatique, l'identification des liens précis de causalité, très difficile pour une machine, se révèle en général assez naturelle pour le concepteur humain. Aussi est-il préférable de mettre à contribution le savoir-faire et l'intuition de ce dernier, dans la mesure où il semble le plus apte à fournir le bon niveau d'information.

Le but de l'invention est de pallier les insuffisances des systèmes de l'art antérieur décrit ci-dessus au moyen d'un procédé adapté aux plateformes de services, dont le résultat principal est la production automatique d'un modèle de performance, à partir d'un modèle fonctionnel du système étudié. Le modèle de performance généré est dédié aux simulateurs logiciels du marché basés sur les systèmes de files d'attente. Le modèle fonctionnel s'exprime sous forme de diagrammes de séquences assortis de données temporelles. Les diagrammes de séquence peuvent être dérivés d'une modélisation dynamique et complète du système faite en SDL ou en UML.

Exposé de l'invention

Le procédé selon l'invention comporte les étapes suivantes :

- Répartir les requêtes représentatives du système en un nombre fini de groupes et identifier, pour chaque groupe de requêtes, le flot d'exécution correspondant, la répartition desdites requêtes étant
5 déterminée par le service invoqué et par les caractéristiques du comportement propre du client, et le flot d'exécution pour chaque groupe de requêtes correspond à l'enchaînement d'exécutions d'entités logicielles, en séquence et/ou en parallèle, induit par
10 une requête du groupe.

- Formaliser les flots d'exécutions à l'aide d'une notation permettant de mettre en évidence, d'une part, les relations causales entre les différentes entités logicielles du système impliquées
15 dans les flots d'exécution, et d'autre part, les informations caractérisant la consommation des ressources du système, telles que la durée d'occupation de CPU lorsqu'une entité logicielle est active.

- Elaborer un modèle intermédiaire comportant en plus des flots d'exécution formalisés, une spécification de ressources décrivant les matériels physiques du système et une spécification de
20 l'environnement représentant le comportement des utilisateurs.

- Automatiser la transformation du modèle intermédiaire élaboré en un modèle de performance.
25

Préférentiellement, le modèle de performance dérivé du modèle intermédiaire élaboré est dédié aux simulateurs logiciels pré-existant utilisant des
30 techniques de réseaux de files d'attente.

Selon une caractéristique de l'invention, la répartition des requêtes du système en un nombre fini de groupes de requêtes est déterminée par le service invoqué (sa nature), et par les caractéristiques du comportement propre du client qui influencent la manière dont se réalise le service invoqué. Le flot d'exécution pour chaque groupe de requêtes est déterminé par l'enchaînement d'exécutions d'entités logicielles, en séquence et/ou en parallèle, induit par une requête du groupe.

Selon l'invention, la topologie du modèle à files d'attente dérivée de la transformation est entièrement déterminée par les flots d'exécution correspondant aux groupes de requêtes.

Selon l'invention, la dérivation d'un modèle de performance dédié à un simulateur pré-existant basé sur des techniques des réseaux de files d'attente est automatisable par adaptation des règles de correspondance proposées.

Selon un mode de réalisation, le formalisme des phases est réalisé à l'aide d'une extension du formalisme MSC (Message Sequence Charts), et la formalisation du graphe des phases et des flots d'exécution d'un service à l'aide du formalisme HMSC (High level Message Sequence Charts) est représentée sous la forme d'un arbre comportant :

- une pluralité de nœuds représentant les phases constituant le service ;
- au moins un arc orienté menant d'un nœud à un autre représentant l'enchaînement en séquence de deux phases ;

L'arbre de formalisation peut comporter en outre :

- au moins un nœud suivi de plusieurs arcs orientés en parallèle,

5 - au moins un nœud suivi de plusieurs arcs orientés en fonction du choix de la phase suivante dépendant soit d'une condition externe au système (liée par exemple à une caractéristique du client), soit d'une condition interne liée à l'état courant du
10 système.

Ainsi, le modèle intermédiaire élaboré comporte les flots d'exécution formalisés, associés à des données temporelles caractérisant le comportement d'entités logicielles et leurs interactions, au moins
15 une spécification des ressources décrivant les matériels physiques, et au moins une spécification d'environnement représentant le comportement des utilisateurs.

Un avantage par rapport au modèle LQN réside
20 dans la disponibilité sur le marché de plusieurs ateliers de simulation industriels, à la maturité et à la puissance éprouvées. Des outils comme QNAP ou
SES/Workbench proposent des fonctionnalités d'analyse
avancées permettant de modéliser une très grande
25 variété de systèmes complexes, ainsi que la possibilité de configurer très finement le modèle, par exemple, les politiques d'ordonnancement des tâches peuvent être redéfinies de façon algorithmique, si aucune ne s'avère suffisamment réaliste parmi les politiques prédéfinies.

Brève description des dessins

- D'autres caractéristiques et avantages de l'invention ressortiront de la description qui va suivre, prise à titre d'exemple non limitatif en référence aux figures annexées dans lesquelles :
- la figure 1 illustre schématiquement le procédé selon l'invention ;
 - la figure 2 illustre schématiquement un modèle intermédiaire défini par le procédé selon l'invention ;
 - la figure 3 illustre schématiquement l'association des différents éléments du modèle intermédiaire et des primitives du simulateur SES/Workbench basé sur des systèmes à files d'attente ;
 - les figures 4a à 4h illustrent schématiquement les règles de correspondance entre les événements annotés caractérisant le comportement d'entités logicielles et des primitives du simulateur SES ;
 - la figure 5 illustre schématiquement un sous-modèle SES dérivé d'un flot d'exécution formalisé en appliquant des règles de correspondance illustrées dans les figures 4a à 4h ;
 - la figure 6 représente un sous-modèle SES de l'architecture globale du système obtenu en appliquant des règles de correspondance illustrées dans la figure 3 ;
 - la figure 7 illustre schématiquement l'arbre des phases représentant les différentes étapes pour la réalisation d'un service Audio-conférence selon

le procédé de l'invention ;

- les figures 8 à 10 représentent les flots d'exécution extraits de l'arbre des phases pour le service Audio-conférence ;

5 - les figures 11 à 20 représentent des schémas de formalisations des phases du comportement du service fourni ;

 - La figure 21 représente la déclaration des ressources, les flots d'exécutions et le sous-
10 modèle de l'architecture globale d'une plate-forme d'audioconférence dans le modèle SES ;

 - La figure 22 représente un sous-modèle SES de l'architecture globale de la plate-forme d'audio-conférence ;

15 - Les figures 23 à 25 représentent les sous-modèles SES dérivés des différents flots d'exécution formalisés en appliquant des règles de correspondance illustrées dans les figures 4a à 4h.

20 Exposé détaillé de modes de réalisation particuliers

Pour modéliser un système étudié, le procédé selon l'invention utilise les systèmes à files d'attente. Ce choix est lié au fait que les simulateurs de performances les plus avancés du marché exploitent
25 pour la plupart le modèle des réseaux de files d'attente. Ces ateliers font preuve d'une maturité notoire, tant sur le plan de la richesse des fonctionnalités, que de la qualité de l'analyse et de la robustesse.

30 Rappelons qu'un modèle d'un système à files d'attente se concentre sur les aspects qui ont un effet

direct ou indirect sur la consommation des ressources du système (processeurs et mémoires des machines, bande passante des réseaux de transport).

Le procédé selon l'invention permet de
5 modéliser essentiellement :

- la répartition des entités logicielles dont la coopération réalise le service sur les différentes machines;
- les interactions, ou relations de
10 causalité, qui existent entre ces entités logicielles,
- les durées de traitement unitaire,
- les politiques d'ordonnancement des tâches concurrentes,
- la capacité des réseaux sous-jacents
15 (caractérisée par les délais de transmission et la politique de routage),
- le comportement des utilisateurs caractérisé par le débit d'arrivée des requêtes, et par la distribution de la durée qui sépare deux arrivées
20 consécutives.

Un système logiciel réparti se compose généralement d'un ensemble d'entités logicielles (processus, composants, etc.) s'exécutant sur un ensemble de machines connectées par des réseaux de
25 transport. Ce type de système se modélise en associant un serveur muni d'une file d'attente à chaque couple composé d'une entité logicielle et d'une machine sur laquelle s'exécute l'entité. Deux entités logicielles hébergées sur une même machine sont considérées comme
30 deux serveurs distincts. Pour chaque serveur ainsi défini, son service et sa file d'attente correspondent

respectivement à l'exécution de l'entité et à la mémoire de la machine.

Les clients sont les requêtes envoyées par les utilisateurs invoquant des services fournis par le système. Quant à la topologie du système à files d'attente, elle est déterminée entièrement par les traces d'exécution. En effet l'arrivée d'une requête dans le système provoque l'exécution d'actions sur les serveurs. Ces actions peuvent évoluer simultanément (en parallèle) ou se synchroniser en séquence. Chaque trace d'exécution est ainsi représentée par une arborescence de serveurs. En définitive, la topologie globale du réseau de files d'attente est telle que toutes les traces d'exécution sont couvertes.

Avant de procéder à la simulation du modèle, celui-ci doit encore être complété d'un modèle d'environnement et d'un modèle de ressources. Le premier caractérise le comportement des utilisateurs, en termes de débit et de distribution statistique des temps d'inter-arrivées. Le second précise la durée d'exécution de chaque entité lorsque la machine lui est entièrement dédiée (tests unitaires), les délais de transmission caractérisant la capacité des réseaux et la politique d'ordonnancement des traitements concurrents. Ces données peuvent être intégrées dans le modèle fonctionnel du système. La simulation consiste à injecter un très grand nombre de requêtes dans le modèle. On optimise l'utilisation des ressources du système en cherchant une bonne configuration (ou paramétrage) des différents éléments du modèle. La qualité de la configuration globale s'évalue en

étudiant par simulation son impact sur la charge des serveurs, sur le taux d'occupation des tampons, et sur des quantités caractérisant la qualité de service: les temps de réponse d'un composant particulier ou du système (de bout en bout), la capacité de traitement maximum (volumétrie), etc.

Pour des logiciels répartis quelconques, l'identification des interactions ou relations de causalité entre les entités logicielles peut être très complexe, voire impossible. En fait, la démarche proposée par l'invention est applicable aux systèmes dont on peut raisonnablement résumer le comportement par un ensemble restreint de scénarios d'exécution. Idéalement, les scénarios choisis évolueront indépendamment les uns des autres. Néanmoins, les systèmes impliquant des dépendances entre scénarios sont acceptables (en cas d'interaction entre deux clients par exemple), tant que ces dépendances demeurent clairement exprimables par le concepteur.

Les plates-formes de services constituent en ce sens un champ d'application concret pour la mise en œuvre de l'invention. En effet, l'utilisation de ce type de plate-forme peut souvent être caractérisée par un nombre raisonnable de scénarios, qui correspondent aux différents cas d'utilisation de la plate-forme (autrement dit, au traitement des différentes requêtes émises par les utilisateurs lors de l'invocation des services), et qui dépendent faiblement les uns des autres.

Comme cela est illustré schématiquement par la figure 1, le véritable point de départ, pour la

génération d'un modèle de performance à files d'attente, consiste à répartir les requêtes représentatives du système en un nombre fini de groupes de requêtes et à identifier (étape 2), pour chaque

5 groupe de requêtes, le flot d'exécution correspondant. Ceci permet de déterminer la topologie globale du système à files d'attente afin que s'y reflètent toutes les traces d'exécution induites par des requêtes représentatives. Les requêtes représentatives sont

10 celles dont la fréquence d'émission dans le système est significative. La répartition de ces requêtes est déterminée d'une part par le service invoqué ; et d'autre part, par les caractéristiques du comportement propre du client qui influencent la manière dont se

15 réalise le service invoqué (voir dans l'exemple d'Audioconférence). Quant au flot d'exécution pour chaque groupe de requêtes, il correspond à l'enchaînement d'exécutions d'entités logicielles, en séquence et/ou en parallèle, induit par une requête du

20 groupe.

Répartition des requêtes et identification des flots d'exécution :

Pour la première étape (n°2 sur la figure 1),

25 chaque service du système est représenté sous la forme d'un arbre aux caractéristiques suivantes :

- les nœuds de l'arbre représentent les phases constituant le service. Un nœud particulier, le nœud initial, correspond à la phase qui initie le

30 service ;

- un arc orienté menant d'un nœud à un

autre représente l'enchaînement en séquence de deux phases

- lorsque plusieurs arcs sont issus d'un même nœud N, cela correspond soit à l'exécution en parallèle de toutes les phases suivantes, soit à un choix de l'exécution d'une et une seule des phases suivantes. Dans ce dernier cas, le choix de la phase suivante est lié à une condition portant soit sur une propriété liée au comportement de l'utilisateur, la condition est alors dite externe, soit sur l'état courant du système, dans ce cas, la condition est dite interne.

Les requêtes sont naturellement regroupées sur la base des choix effectués au niveau des conditions externes. Les flots d'exécution correspondants peuvent être extraits de l'arbre global en parcourant celui-ci à partir du nœud initial, et en ne retenant qu'un seul successeur à chaque condition externe rencontrée. Par conséquent, un flot d'exécution est un sous-arbre de l'arbre global caractérisé par la transformation systématique de tout alternative externe parmi N successeurs en un enchaînement en séquence vers un seul de ces successeurs. Lors du parcours, tout autre aspect structurel de l'arbre est en revanche conservé (enchaînement en séquence vers un nœud simple, enchaînement parallélisé vers plusieurs nœuds en simultané, enchaînement conditionné par l'état interne du système). Les figures 8 à 10 illustrent les flots d'exécution extraits de l'arbre des phases (figure 7) dans l'exemple du service Audio-conférence.

Formalisation des flots d'exécution

L'étape 4 du procédé (figure 1) consiste à formaliser les flots d'exécution à l'aide d'une notation permettant de mettre en évidence les interactions entre les différentes entités logicielles impliquées dans les flots d'exécution et les informations caractérisant la consommation des ressources du système.

Dans le mode de réalisation décrit, cette formalisation est basée sur les Message Sequence Charts (MSC). Cependant, d'autres formalismes de diagrammes de séquence pourraient être utilisés. La cohérence d'un MSC est optimale si le diagramme est produit en tant que trace d'exécution du système. En phase de conception, une spécification exécutable du système peut-être produite en langage SDL, qui se trouve être un formalisme étroitement lié aux MSC sur le plan de la sémantique. En pratique, tous les outils SDL sont capables de produire des traces de simulation dans le format MSC. Ces mêmes outils offrent en outre la possibilité d'éditer manuellement de tels diagrammes, et c'est ainsi que l'on pourra procéder à défaut d'exploiter la simulation d'un modèle SDL.

Les évènements qui apparaissent sur un diagramme MSC portent essentiellement sur des aspects fonctionnels : émission ou réception de message, action, armement ou expiration de temporisation, etc. En réalité, ce niveau d'information est non pertinent ou insuffisant pour un modèle de performance : par exemple, le nom des messages n'a pas d'importance. Pour

Ces quatre types d'information sont indispensables au modèle de performance, pour calculer le temps de réponse, la répartition de charge des machines, la longueur des tampons, etc. Le procédé définit cinq clauses correspondantes (dont deux pour les conditions d'enchaînement), qui constituent ainsi l'extension de la notation pour formaliser les flots d'exécution : EXEC (occupation du processeur), EXPECT (condition d'enchaînement sur une expression booléenne), DELAY (condition d'enchaînement sur l'écoulement du temps), PUBLISH (publication de faits), et END (terminaison de la branche du flot). Les figures 11 à 20 illustrent les flots d'exécution formalisés dans l'exemple d'un modèle fonctionnel du service Audioconférence.

Elaboration du modèle intermédiaire

L'étape 6 du procédé (figure 1) consiste à définir un modèle intermédiaire destiné à permettre le passage automatique à un modèle de performance. Sur la figure 2, on voit que ce modèle est composé des flots d'exécution formalisés (10) caractérisant le comportement d'entités logicielles et leurs interactions, d'une spécification des ressources (12) décrivant les matériels physiques, et d'une spécification d'environnement 14 représentant le comportement des utilisateurs.

Le modèle de ressources contient les déclarations suivantes :

- les machines, la taille des tampons et la politique d'ordonnancement des traitements simultanés

de ces machines ;

- les réseaux de transport et leurs topologies permettant la connexion des machines ;

- les entités logicielles (programmes à unique fil d'exécution), ainsi que leur répartition sur les ressources ;

- les capacités des machines exprimées par rapport aux temps de traitement unitaires des tâches (formalisées par EXEC) ;

- les capacités des liens de transmission exprimées par rapport aux délais de transmission ;

Dans le modèle d'environnement, qui caractérise le trafic de requêtes généré par les utilisateurs, il faut préciser les données suivantes :

- le débit d'arrivée des requêtes ;

- le rythme d'arrivée ; il est caractérisé entièrement par la distribution de la durée séparant deux arrivées consécutives ;

- les attributs ; ce sont des variables propres à chaque requête ; ils permettent notamment d'identifier les différentes classes de requêtes déterminées par les flots d'exécution ; d'autres variables servent à exprimer les conditions booléennes de synchronisation entre des branches parallèles d'un même flot d'exécution.

- la proportion de chaque classe de requêtes (cette répartition est fournie par des observations ou des estimations quant à l'usage futur de la plate-forme)

Transformation automatique en un modèle de performance L'étape 8 du procédé (figure 1) consiste à définir des règles de correspondance entre les éléments du modèle intermédiaire et les primitives d'un simulateur généraliste basé sur des systèmes à files d'attente. La transformation devient automatique lorsqu'on applique ces règles de correspondance. Ces dernières s'appliquent aussi bien à tous les simulateurs de cette catégorie : tous sont fondés sur les mêmes concepts de base, seul le mode de représentation de ces concepts diffère. La cohérence de la transformation repose sur des règles de sémantique (non explicitées dans le présent document), qui permettent de détecter les spécifications erronées conduisant à des modèles de performance incomplets ou dénués de sens. Nous illustrons ici la démarche dans le cas d'utilisation du simulateur SES/Workbench.

La figure 3 illustre schématiquement les règles de correspondance entre le modèle intermédiaire et des « nœuds » SES/Workbench. En désignant respectivement par $R = \{\text{machine}_1, \dots, \text{machine}_m\}$ et $F = \{\text{flotExecFormalisé}_1, \dots, \text{flotExecFormalisé}_n\}$ l'ensemble des machines du système déclarées dans le modèle de ressources et l'ensemble des flots d'exécution formalisés des services du système :

- Le modèle d'environnement 14 est associé à un nœud « source » 20, suivi éventuellement d'un nœud « submodel » 22. Ce dernier est utile dans les cas plus complexes, par exemple dans le cas de requêtes structurées par sessions ou à plusieurs niveaux ;

- Le modèle de ressources 12 contenant la

déclaration de l'ensemble R de m machines est associé à un ensemble de m nœuds « service » dénotés par un nom logique (serveur_j par exemple), pour tout j ;

5 - Chaque flotExecFormalisé_i 10 est associé à un nœud « submodel » dénoté par un nom logique (flotExec_i par exemple) pour tout i ;

10 - Chaque nœud « submodel » associé au flot d'exécution formalisé est développé, en faisant les correspondances suivantes entre évènements du flot d'exécution formalisé et primitives SES. Les différentes associations sont décrites par les figures 4a à 4h.

En référence à la figure 4a, la primitive « EXEC(t) » est associée à une séquence de deux nœuds :
15 un nœud « user », suivi d'un nœud « service reference_to ». Le nœud user permet de spécifier la valeur de t correspondant à la durée d'exécution de code, tandis que le nœud « service reference_to » fait référence au nœud service déclaré dans le module qui
20 correspond à la machine hébergeant le processus qui effectue le « EXEC(t) ».

La publication de faits « PUBLISH » est associée (figure 4b) à un nœud « user » contenant le programme C adéquat. Ce programme modifie les variables
25 qui représentent les clés ou faits à publier.

La fin de branche de flot « END » est associée à un nœud « sink » (figure 4c).

La primitive « DELAY(t) » est associée à un nœud « delay » (figure 4d) et la valeur de t est le
30 paramètre du nœud.

La condition d'enchaînement « EXPECT » est associée à un nœud « block », où elle s'exprime par une expression booléenne c donnée en paramètre du nœud (figure 4 e).

5 L'enchaînement en séquence des événements est représenté par un arc entre les nœuds (figure 4f).

Le choix d'un enchaînement des événements parmi N enchaînements possibles est représenté par N arcs partant du nœud où le choix a lieu. Les conditions
10 déterminant un choix sont spécifiées dans l'arc correspondant (figure 4g).

Enfin la séparation du flot en N branches parallèles est représentée par un nœud « split » (figure 4h).

15 La figure 5 illustre la séquence correspondant au sous-modèle d'un flot d'exécution formalisé complétée par un nœud « enter » et par un nœud « return » qui représentent respectivement l'entrée et la sortie du sous-modèle.

20 La figure 6 illustre l'architecture globale du système, qui rassemble les sous-modèles flots d'exécution et le sous-modèle d'environnement.

Exemple d'application : plate-forme d'Audio-conférence

25 Présentation informelle du service d'audio-conférence.

Le service d'Audio-conférence permet à un nombre déterminé d'utilisateurs, sur accord préalable, chacun de son côté et à une date et heure convenues, de
30 composer un numéro d'appel afin d'établir une communication vers un pont de conférence. La

réservation d'un pont de conférence ayant préalablement été effectuée par l'un des participants, qui a précisé la date, l'heure, la durée et le nombre de participants.

5 La réalisation du service d'établissement d'une session de conférence, offert par la plate-forme d'Audio-conférence, dépend du rôle (de la nature) du client qui l'invoque :

- est-il organisateur de la session ?
- 10 - est-il un simple participant ?
- est-il autorisé à participer à la session ? Cette classification permet d'identifier les différents types de requêtes. Par ailleurs, il se peut qu'un type de requête interagisse avec un autre. Par exemple, le
- 15 traitement de la requête d'un simple participant à une session d'Audio-conférence est suspendu jusqu'à l'arrivée de l'organisateur de cette session.

Un rôle particulier est affecté à l'Organisateur de la conférence. La conférence ne peut

20 commencer avant l'arrivée de l'Organisateur, c'est-à-dire que les participants ne peuvent pas être connectés au pont de conférence tant que l'Organisateur est absent. La conférence est fermée après raccrochage de

25 l'Organisateur, et les participants encore en ligne seront alors déconnectés du pont de conférence. La souscription, qui consiste principalement en la réservation d'un numéro d'accès au pont de conférence, s'opère par l'intermédiaire du fournisseur de service. Les données sont en conséquence récupérées dans la base

30 de données.

Traitement d'appel

Le traitement d'appel se décompose suivant les phases suivantes (remarque : on ne s'intéresse, pour cette étude de cas, qu'à l'ouverture de conférences, et non à leur évolution ni à leur clôture) :

1) Un participant compose le numéro d'accès à la conférence programmée (numéro d'accès au service NAS). Ce numéro est composé d'un préfixe de type OZABPQ spécifique au service, tandis que le suffixe MCDU est spécifique à la conférence souscrite. Le service effectue la traduction du numéro d'accès au service en numéro réel d'accès au pont de conférence (NAC).

2) Plusieurs cas sont possibles:

- Tant que le nombre de participants est inférieur à la capacité maximale du pont de conférence et que l'Organisateur n'a pas invoqué le service, un message est émis qui annonce le retard de l'organisateur, puis un second invitant à attendre l'ouverture de la conférence.

Quand l'Organisateur se présente, les participants en attente sont alors connectés au pont de conférence (ainsi que l'Organisateur). La conférence est dite ouverte.

- Si le nombre de participants déjà connectés atteint le nombre maximal défini par le souscripteur, le service rejette l'appel avec un message de saturation.

Messages spécifiques au service

Les libellés des messages spécifiques au service sont les suivants :

- accueil : "vous avez composé un numéro d'accès à une

conférence, veuillez ne pas quitter".

- saturation : "le numéro composé est saturé, nous ne pouvons donner suite à votre appel".

5 - incident : "suite à un incident, nous ne pouvons donner suite à votre appel".

- annonce : "votre organisateur n'est pas encore connecté...".

10 - attente : "veuillez ne pas quitter, vous serez mis en communication avec vos correspondants dès ouverture de la conférence...".

Données

Les données spécifiques au service sont :

- les n° d'accès au service
- 15 - les n° traduits (le numéro réseau du pont)
- le nombre maximal de participants par pont de conférence
- l'état de la ressource associée au pont de conférence
- la table des annonces
- 20 - les tables d'analyse nationale, internationale et spéciale (pour des contrôles éventuels sur le numéro appelant).

Ces données sont de différents types, suivant que leur modification se fait ou non par les scripts du traitement d'appel. Les données non modifiables par le traitement d'appel sont dans une base de données appelée ici SDF (Service Data Function), elles sont accessibles en lecture par les scripts du traitement d'appel et en écriture pas des scripts de gestion. Les données modifiables par le traitement d'appel sont dans une base de données accessible en lecture et écriture

par les scripts du traitement d'appel, cette base de données doit fournir des performances d'accès plus importantes. Ce dernier type de donnée est représenté ici par le nombre maximal de participants, et par
5 l'état de la ressource.

Application des étapes du procédé

A partir de la description informelle du service, la première étape consiste à identifier les
10 phases élémentaires et leur organisation en un arbre de phases.

Tous les comportements des utilisateurs ne sont pas pris en compte, seuls sont exprimés les cas d'utilisation les plus représentatifs. En particulier,
15 les raccrochages sont pris en compte en phase de conversation et en phase d'attente, cas de raccrochage les plus pertinents d'un point de vue utilisateur.

La figure 7 illustre l'arbre global des phases qui décrit l'enchaînement des différentes phases en
20 précisant les alternatives internes ou externes. On reconnaît les conditions externes au fait que les arcs qui en sont issus portent des pourcentages statistiques.

Après une phase de connexion P1, une
25 alternative externe permet d'identifier si l'utilisateur est l'organisateur (P2) ou un simple participant (P4). Si l'utilisateur est un participant, une alternative interne {nbParticipants<MAXI}, dont l'option dépend de l'exécution d'autres scénarios,
30 identifie si le nombre maximum de participant est atteint ou non. Si ce nombre est atteint, une phase

pour l'envoi d'un message de saturation est initiée (P6). Si ce nombre maximum n'est pas atteint (P5), une deuxième alternative interne {organisateurPresent} permet d'identifier si l'organisateur est déjà présent.

5 S'il est déjà présent, le participant est connecté au pont (P7, avec en préalable un message d'accueil). Si l'organisateur est absent, un message d'attente est émis vers le participant (phase P8). De cette phase, une condition externe {participantPatient} permet de

10 distinguer le cas où le participant attend l'arrivée de l'organisateur (phase P10) et entre dans la conférence (phase P7) à l'arrivée de l'Organisateur, du cas où le participant est plutôt du type "impatient", c'est-à-dire qu'il raccroche suite à l'annonce de retard

15 diffusée en phase P9. Dans ce cas, le nombre de participants est diminué d'une unité. Encore une fois, on suppose que seules les performances des phases liées à l'établissement de la conférence nous intéressent dans cet exemple, ce qui explique que l'arbre ne

20 contienne aucune phase liée à l'évolution ultérieure de la conférence.

Les requêtes dans l'exemple sont réparties en trois groupes : requêtes envoyées par un organisateur ; requêtes envoyées par un simple participant patient ;

25 et requêtes envoyées par un simple participant impatient. A partir de l'arbre global des phases, on extrait alors les trois flots d'exécution correspondants en ne gardant qu'une branche de chaque alternative externe :

30 - flot d'exécution induit par l'arrivée de l'organisateur (figure 8) ;

- flot d'exécution induit par l'arrivée d'un simple participant patient (figure 9);
- flot d'exécution induit par l'arrivée d'un simple participant impatient (figure 10).

5

Identification des clés et portes des phases concernées

La deuxième étape du procédé consiste à identifier les clés et portes qui influencent le comportement d'un cas d'utilisation à partir du
10 comportement d'un ou plusieurs autres cas d'utilisation. Dans le cas du service d'Audio-conférence, on identifie une clé liée à la présence de l'organisateur. La phase P3, exécutée lors de l'arrivée de l'organisateur, inclut la
15 tâche PUBLISH(organisateurPrésent) .

Une telle publication permettra de débloquer la phase P10 qui inclut une tâche « EXPECT(organisateurPrésent) ».

La variable nbParticipants est augmentée de 1
20 lors de la connexion d'un utilisateur (non organisateur), sauf si elle vaut déjà sa valeur maximum autorisée (MAXI). Ce test est représenté par une alternative interne dans l'arbre de phases.

25 Formalisation des phases

Les figures 11 à 20 représentent des schémas de formalisation des phases du comportement du service, les MSC associés à différentes phases du comportement du service seront donnés avec une vue simplifiée sur le
30 nom des signaux. De même, tous les échanges ne sont pas indiqués afin de ne pas alourdir les schémas.

Les instances considérées dans ces MSC sont :

- le SDF (Service Data Function ou base des données de service),
- le SCF (Service Control Function ou plate-forme d'exécutions des services),
- le SRF (Service Resource Function ou plate-forme vocale) et les USER (utilisateurs).

Les échanges de signaux correspondent:

- aux interrogations de la base de données :
 - appellant(numéroAppelant) interroge la base sur le numero appellant (P2,P4) ;
 - typeAppelant est la réponse de la base, elle peut prendre les valeurs ORGANISATEUR (si l'appelant est l'organisateur) ou PARTICIPANT (si l'appelant n'est pas l'organisateur).
- aux échanges de messages vocaux en passant par la plate-forme vocale : send_message vers la plate-forme vocale, msg_info vers l'utilisateur (P6, P7, P9).
- aux échanges pour la synchronisation par les portes :
 - PUBLISH(organisateurPresent) permettra de publier le fait que l'organisateur vient de se connecter.
 - EXPECT(organisateurPresent) permet le blocage de l'exécution de l'entité SCF jusqu'à l'arrivée de l'organisateur.

La figure 11 représente la formalisation de la phase d'invocation du service.

- La figure 12 représente la formalisation de la phase « IdentificationOrganisateur ».

La figure 13 représente la formalisation de la phase « IdentificationParticipant. » .

La figure 14 représente la formalisation de la phase « OuvertureDeLaConférence » .

5 La figure 15 représente la formalisation de la phase « ConnexionParticipant » .

La figure 16 représente la formalisation de la phase « DeconnexionSuiteASaturation » .

10 La figure 17 représente la formalisation de la phase » EntreeDansLaConference » .

La figure 18 représente la formalisation de la phase « AnnonceRetardOrganisateur » .

La figure 19 représente la formalisation de la phase « AttenteOrganisateur » .

15 La figure 20 représente la formalisation de la phase « AbandonSurAttente ».

Modèle de performance dérivé du modèle fonctionnel

20 La figure 21 représente la vue générale du modèle intermédiaire (dans SES/Workbench). On y trouve : la déclaration des ressources, les trois flots d'exécutions et le sous-modèle de l'architecture globale de la plate-forme d'Audio-conférence. Dans cet exemple, les processus SDF, SCF et SRF s'exécutent
25 chacun sur une machine.

Une vue interne de l'architecture globale de la plate-forme est représentée par la figure 22.

Le nœud initSession initie la création du nombre maximum de sessions simultanées autorisées par
30 la plate-forme. Le nœud appelsDuneSession génère les appels pour chaque session ouverte et le nœud

interAppOrdre modélise les durées qui séparent les appels consécutifs d'une session et le rang auquel arrive l'organisateur. Enfin, chaque fin de session, représentée par le nœud fermetureSession déclenche la
5 création d'une nouvelle session représentée par le nœud creationSession.

Les Figures 23, 24, et 25 montrent respectivement les trois sous-modèles résultant de la transformation des flots d'exécutions de
10 l'organisateur, d'un participant patient, et d'un participant impatient.

REVENDEICATIONS

1. Procédé de génération d'un modèle de performance à partir d'un modèle fonctionnel d'un système comportant une pluralité d'entités matérielles et logicielles réparties coopérant pour fournir un service à au moins un utilisateur, caractérisé en ce qu'il comporte les étapes suivantes :

- Répartir les requêtes représentatives du système en un nombre fini de groupes et identifier, pour chaque groupe de requêtes, le flot d'exécution correspondant, la répartition desdites requêtes étant déterminée par le service invoqué et par les caractéristiques du comportement propre du client, et le flot d'exécution pour chaque groupe de requêtes correspond à l'enchaînement d'exécutions d'entités logicielles, en séquence et/ou en parallèle, induit par une requête du groupe,

- Formaliser les flots d'exécution à l'aide d'une notation permettant de mettre en évidence, d'une part, les relations causales entre les différentes entités logicielles du système impliquées dans les flots d'exécution, et d'autre part, les informations caractérisant la consommation des ressources du système, telles que la durée d'occupation de CPU lorsqu'une entité logicielle est active.

- Elaborer un modèle intermédiaire comportant en plus des flots d'exécution formalisés, une spécification de ressources décrivant les matériels physiques du système et une spécification de l'environnement représentant le comportement des utilisateurs.

- Automatiser la transformation du modèle intermédiaire élaboré en un modèle de performance.

2.Procédé selon la revendication 1, caractérisé en ce que le modèle de performance dérivé du modèle intermédiaire élaboré est dédié aux simulateurs logiciels pré-existant utilisant des techniques de réseaux de files d'attente.

3.Procédé selon la revendication 1, caractérisé en ce que la répartition des requêtes du système en un nombre fini de groupes de requêtes est déterminée par le service invoqué et par les caractéristiques du comportement propre du client qui influencent la manière dont se réalise le service invoqué.

4.Procédé selon l'une des revendications 1 à 3, caractérisé en ce que le flot d'exécution pour chaque groupe de requêtes est déterminé par l'enchaînement d'exécutions d'entités logicielles, en séquence et/ou en parallèle, induit par une requête du groupe.

5.Procédé selon la revendication 4, caractérisé en ce que la topologie du modèle à files d'attente dérivé de la transformation est entièrement déterminée par les flots d'exécution correspondant aux groupes de requêtes.

6.Procédé selon la revendication 4, caractérisé en ce que la dérivation d'un modèle de performance dédié à un simulateur pré-existant basé sur des techniques des réseaux de files d'attente est automatisable par adaptation des règles de correspondance proposées.

7.Procédé selon l'une des revendications 1 à 6, caractérisé en ce que le formalisme des phases est

réalisé à l'aide d'une extension du formalisme MSC (Message Sequence Charts).

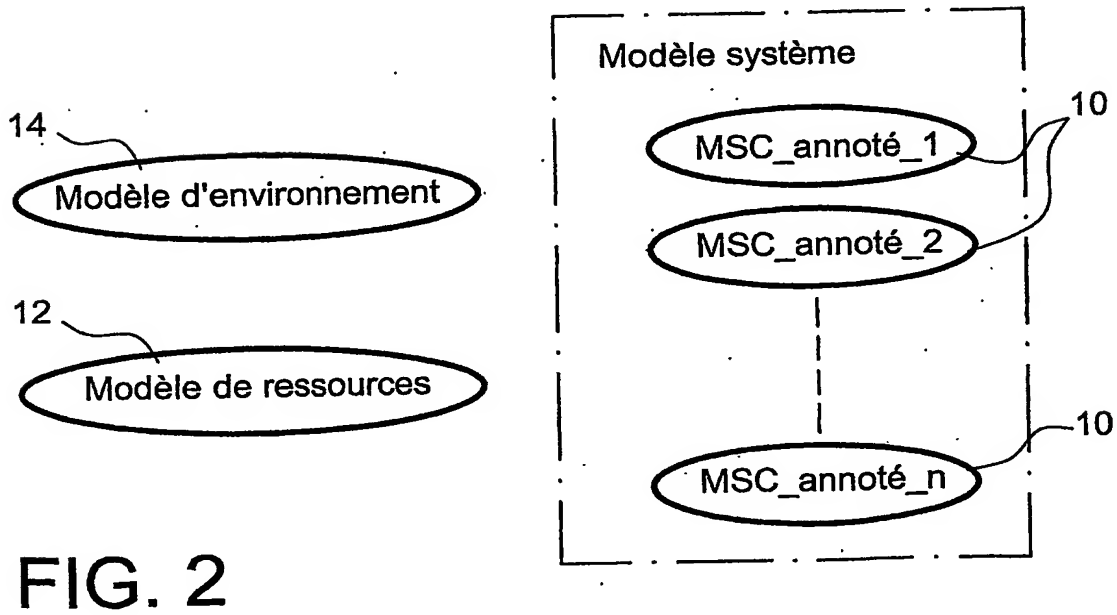
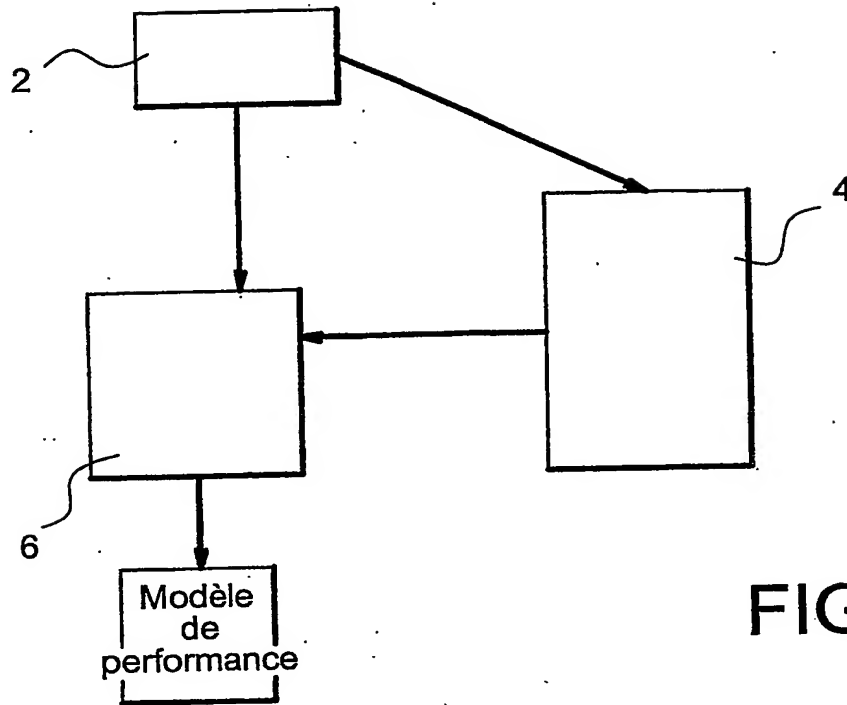
8. Procédé selon l'une des revendications 1 à 7, caractérisé en ce que la formalisation du graphe des phases et des flots d'exécution d'un service à l'aide du formalisme HMSC (High level Message Sequence Charts) est représenté sous la forme d'un arbre comportant :

- une pluralité de nœuds représentant les phases constituant le service ;
- 10 - au moins un arc orienté menant d'un nœud à un autre représentant l'enchaînement en séquence de deux phases ;

9. Procédé selon la revendication 8, caractérisé en ce que l'arbre de formalisation comporte en outre :

- au moins un nœud suivi de plusieurs arcs orientés en parallèle,
- au moins un nœud suivi de plusieurs arcs orientés en fonction du choix de la phase suivante dépendant soit d'une condition externe au système, soit d'une condition interne liée à l'état courant du système.

10. Procédé selon l'une des revendications 1 à 9, caractérisé en ce que le modèle intermédiaire élaboré comporte les flots d'exécution formalisés caractérisant le comportement d'entités logicielles et leurs interactions, au moins une spécification des ressources décrivant les matériels physiques, et au moins une spécification d'environnement représentant le comportement des utilisateurs.



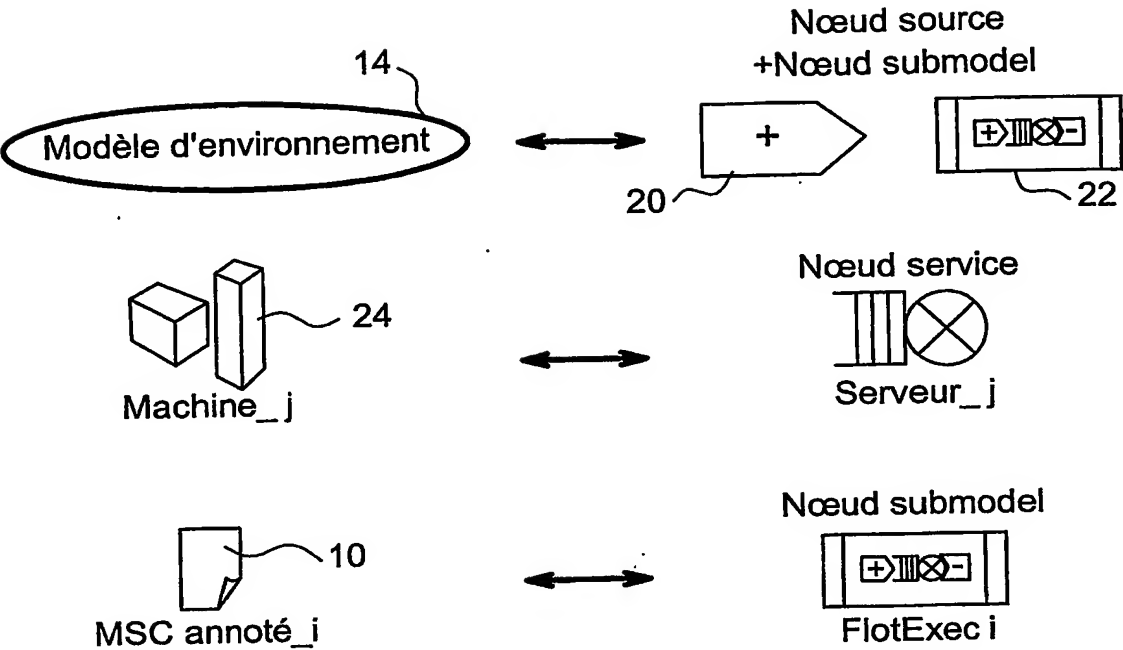
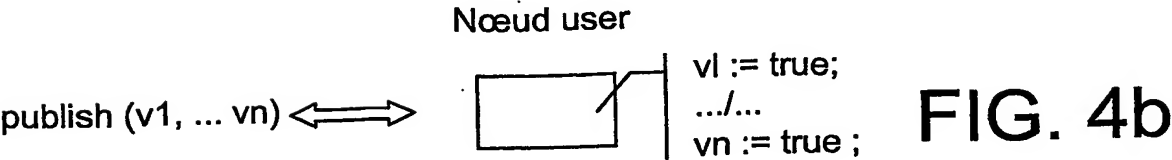
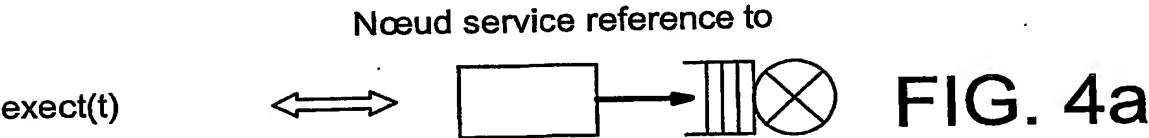
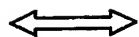


FIG. 3



end



Nœud sink

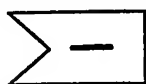


FIG. 4c

delay(t)



Nœud delay

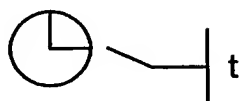
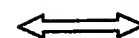


FIG. 4d

expect(c)



Nœud block

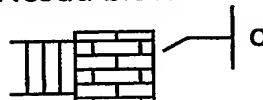
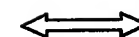


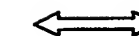
FIG. 4e

enchaînement
en séquence

oriented arc



FIG. 4f

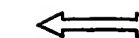
choix d'un
enchaînement

N oriented arcs



FIG. 4g

Emissions



Nœud split

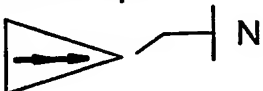


FIG. 4h

submodel MSC_annoté

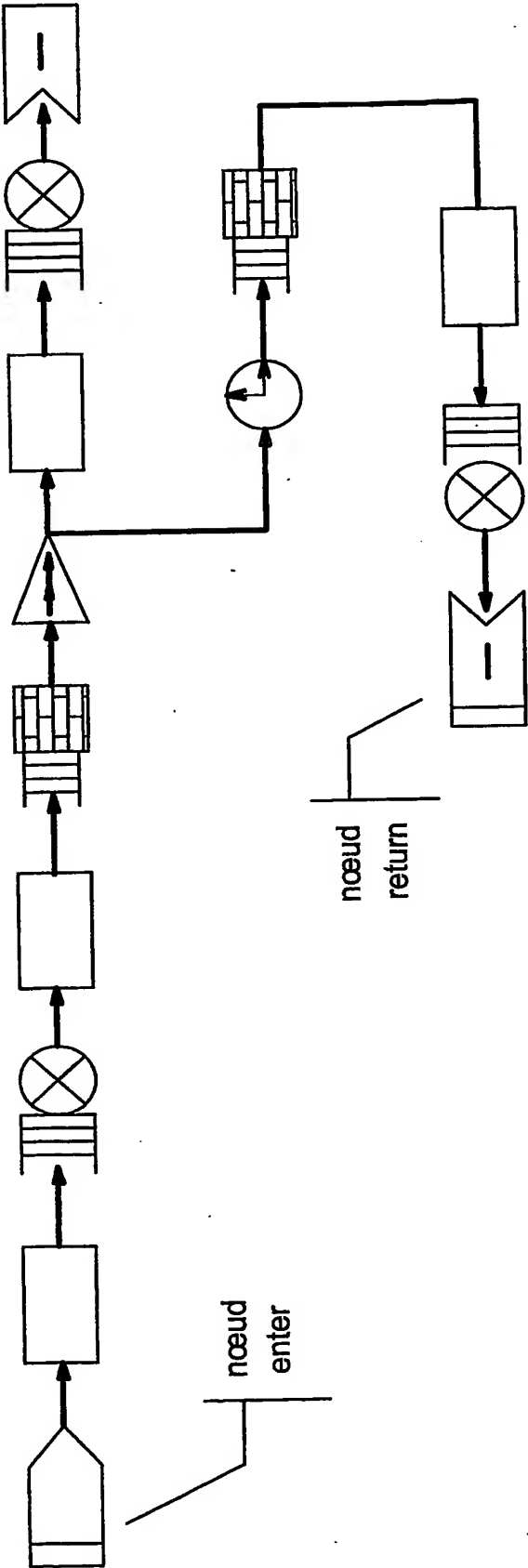
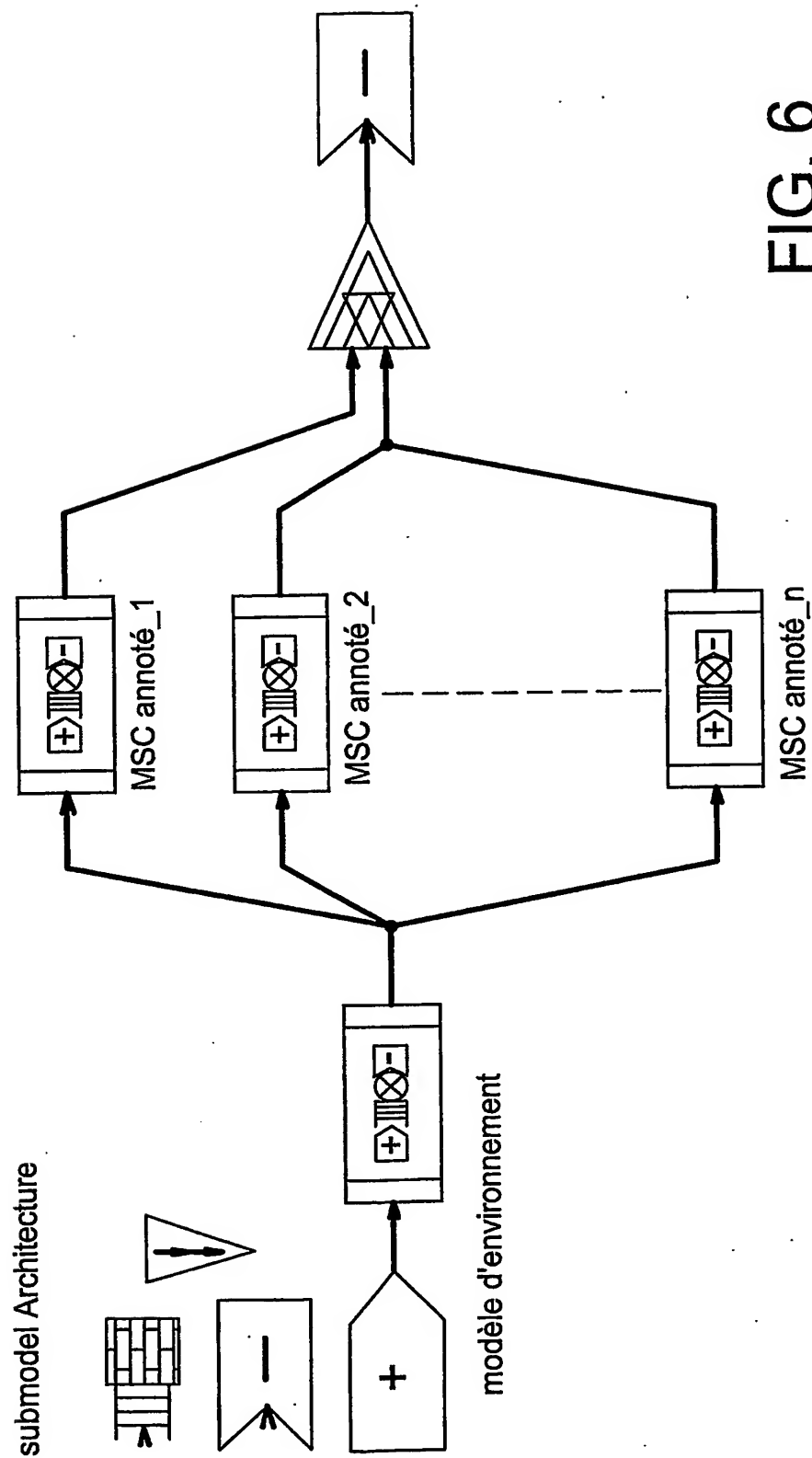


FIG. 5



6 / 17

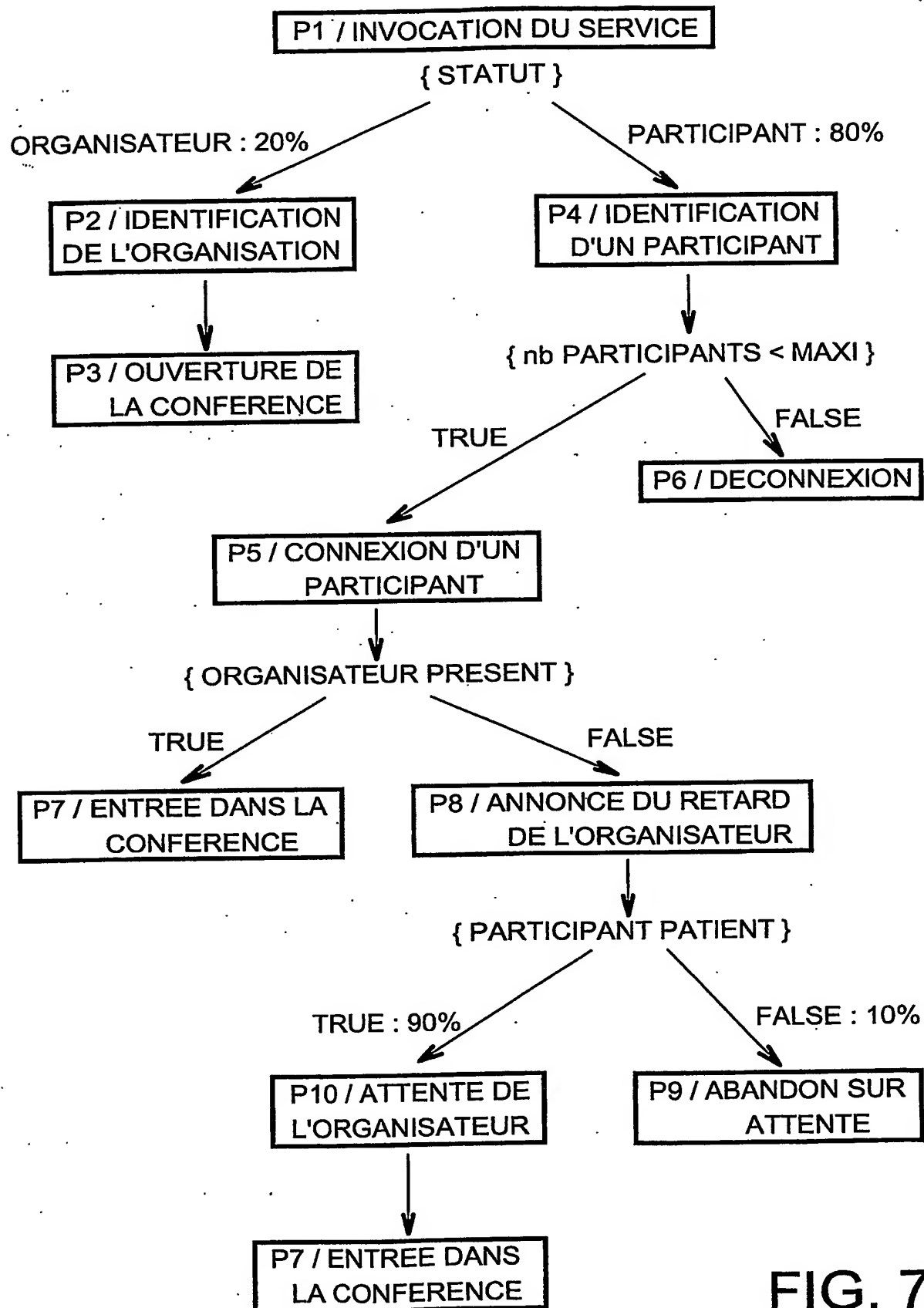


FIG. 7

7 / 17

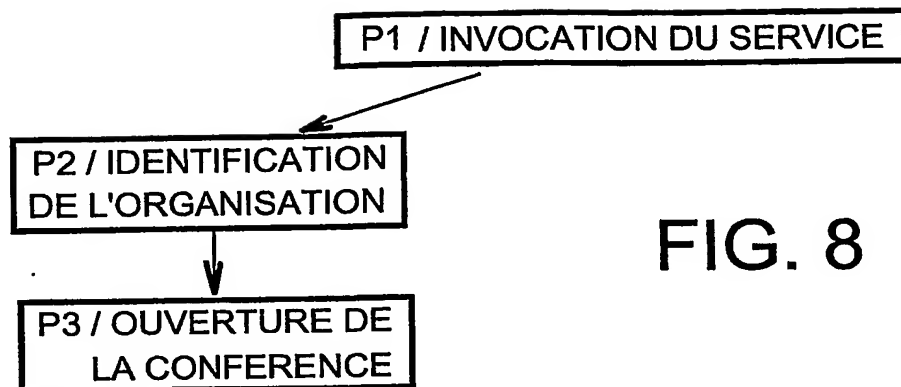


FIG. 8

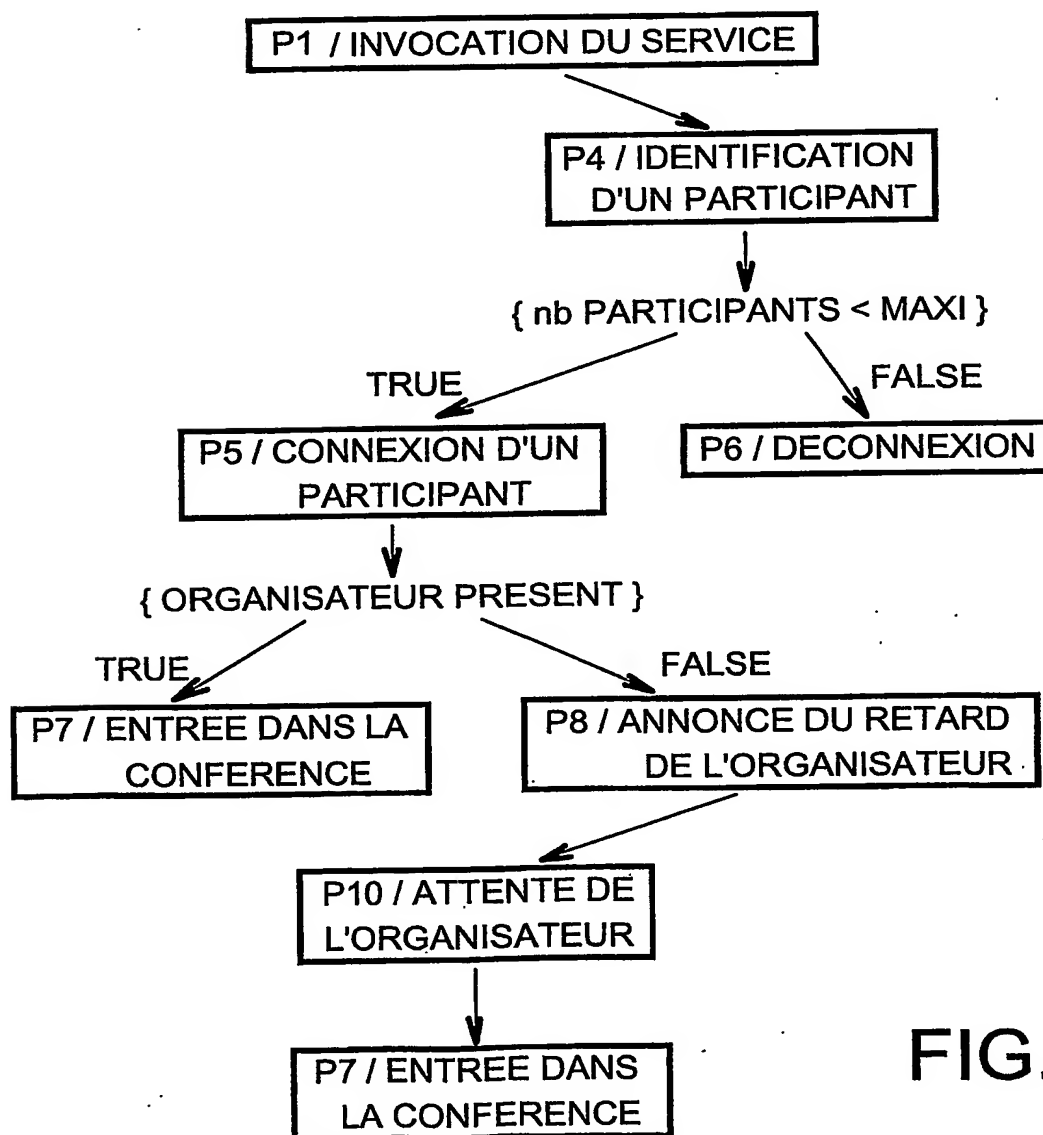


FIG. 9

8 / 17

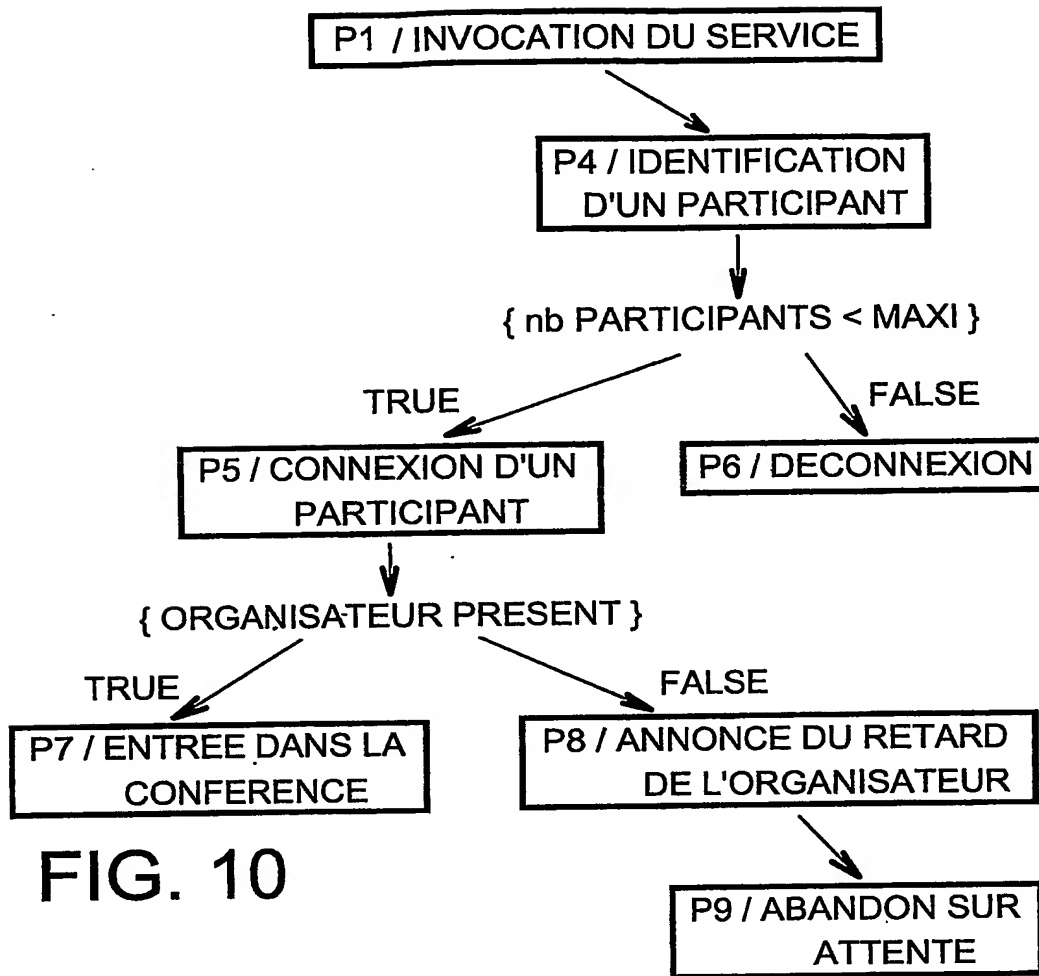
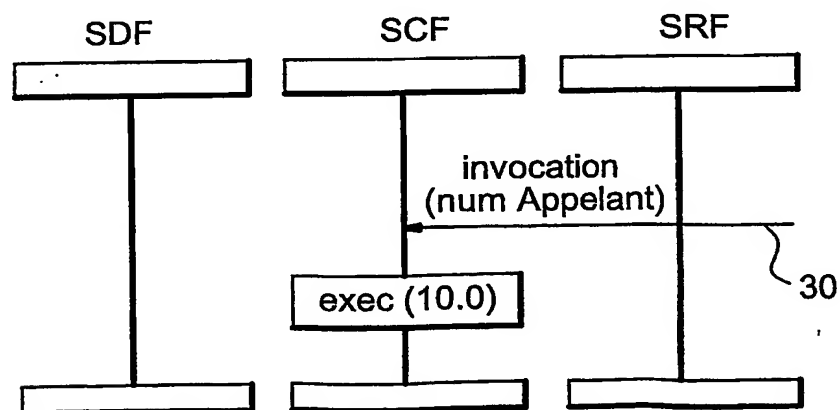


FIG. 11

Invocation Du Service



Identification Organisateur

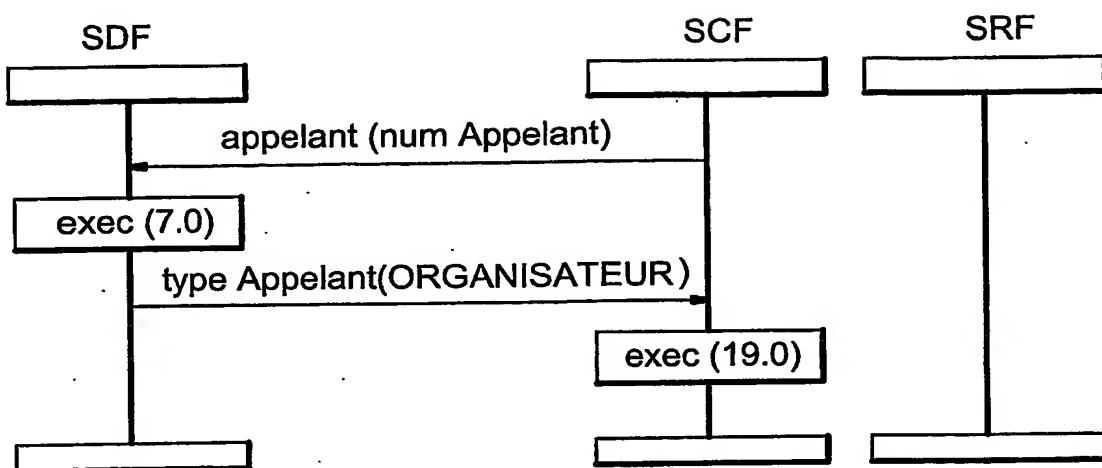


FIG. 12

Identification Participant

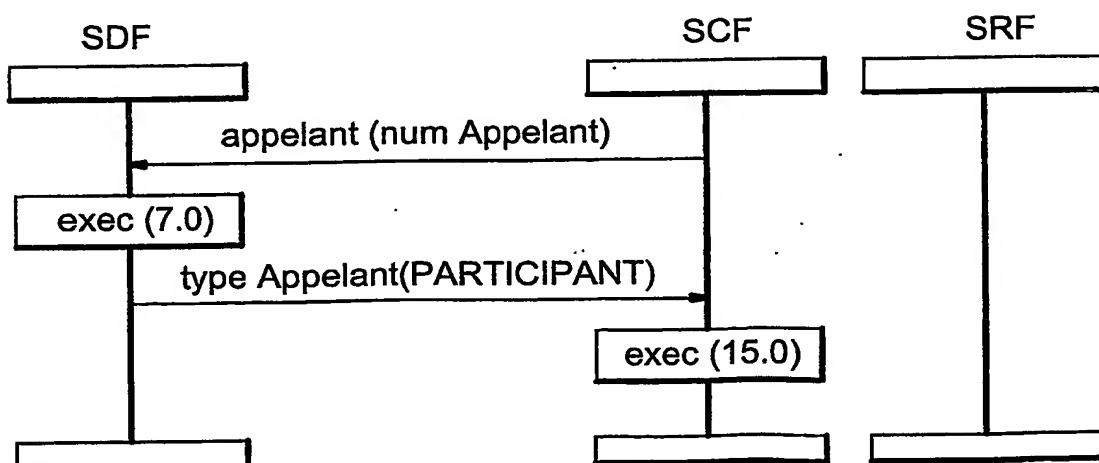


FIG. 13

Ouverture De La Conférence

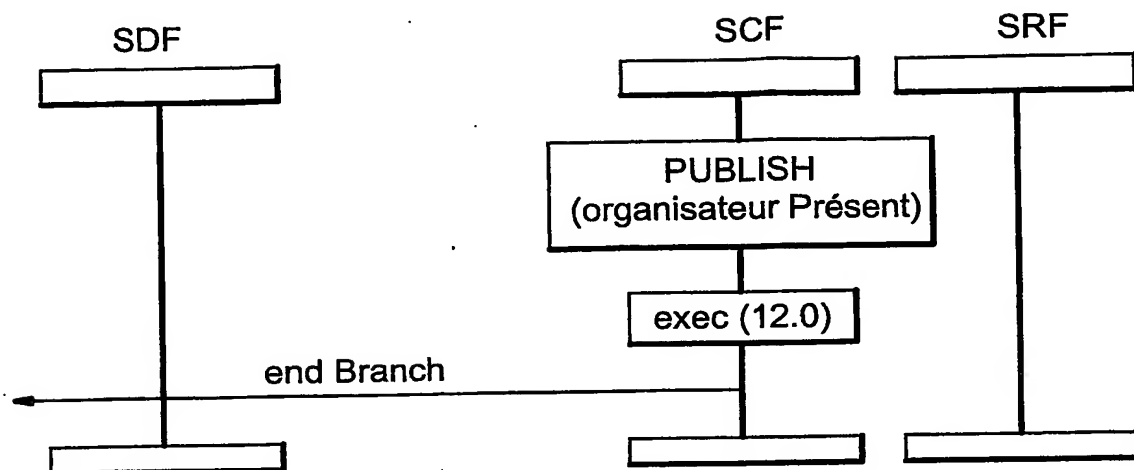


FIG. 14

Connexion Participant

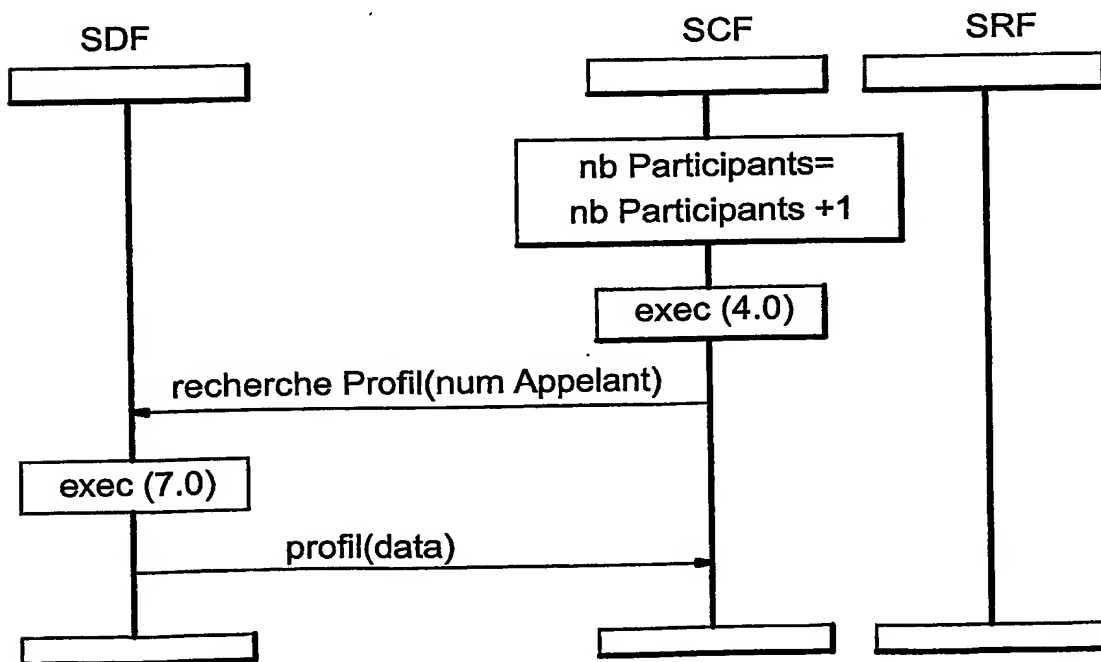
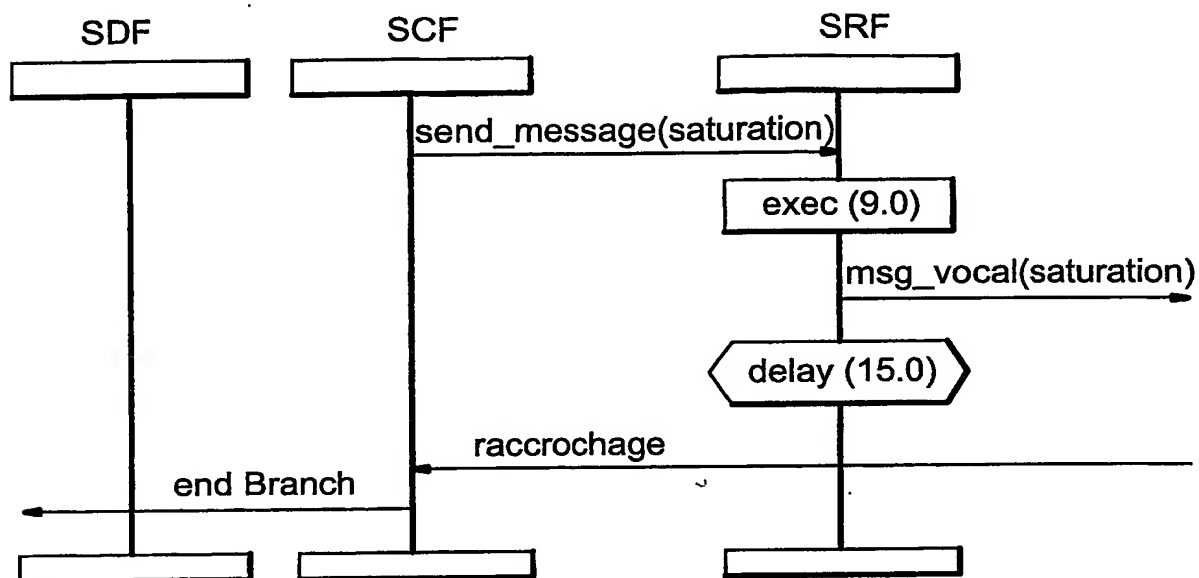


FIG. 15

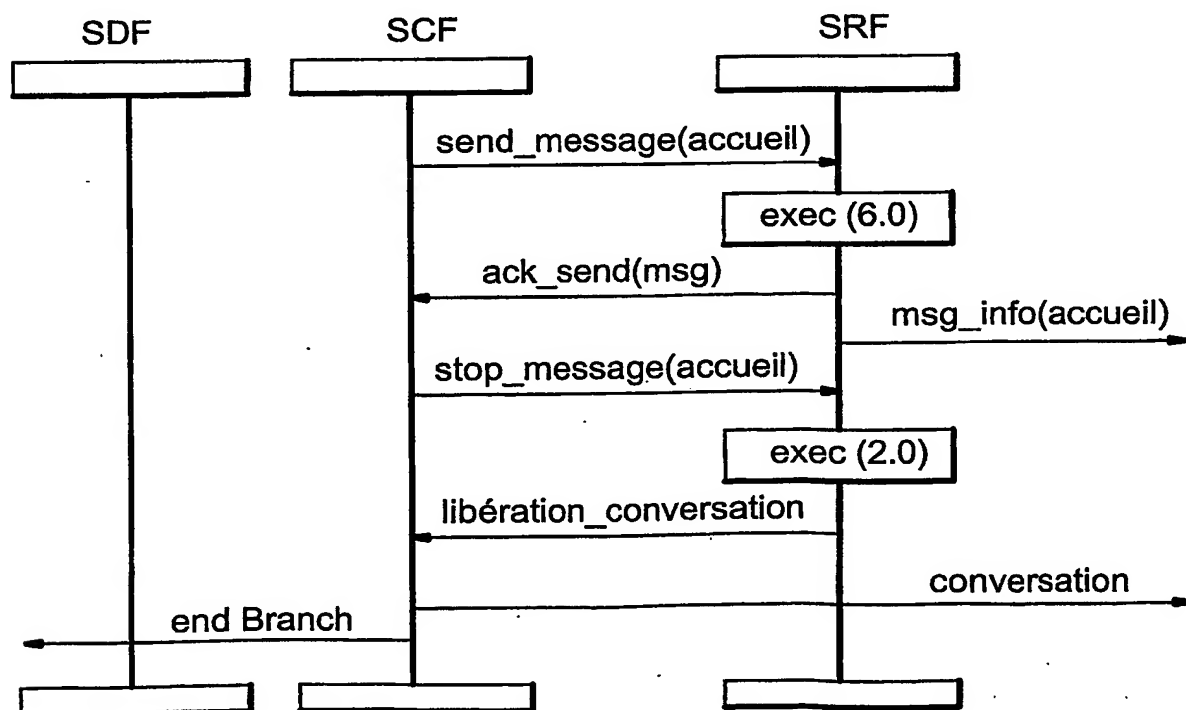
Deconnexion Suite A Saturation

FIG. 16



Entrée Dans La Conférence

FIG. 17



Annonce Retard Organisateur

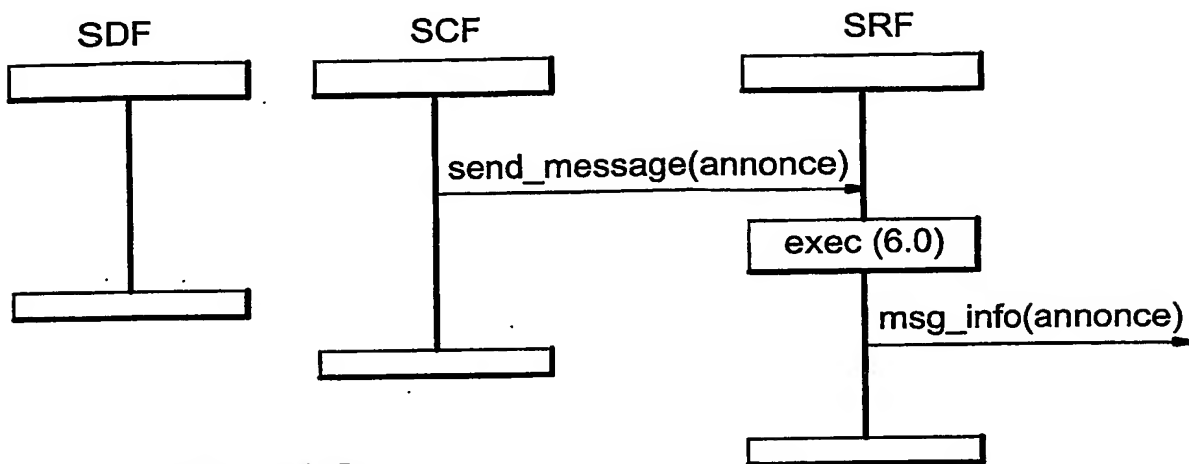


FIG. 18

Attente Organisateur

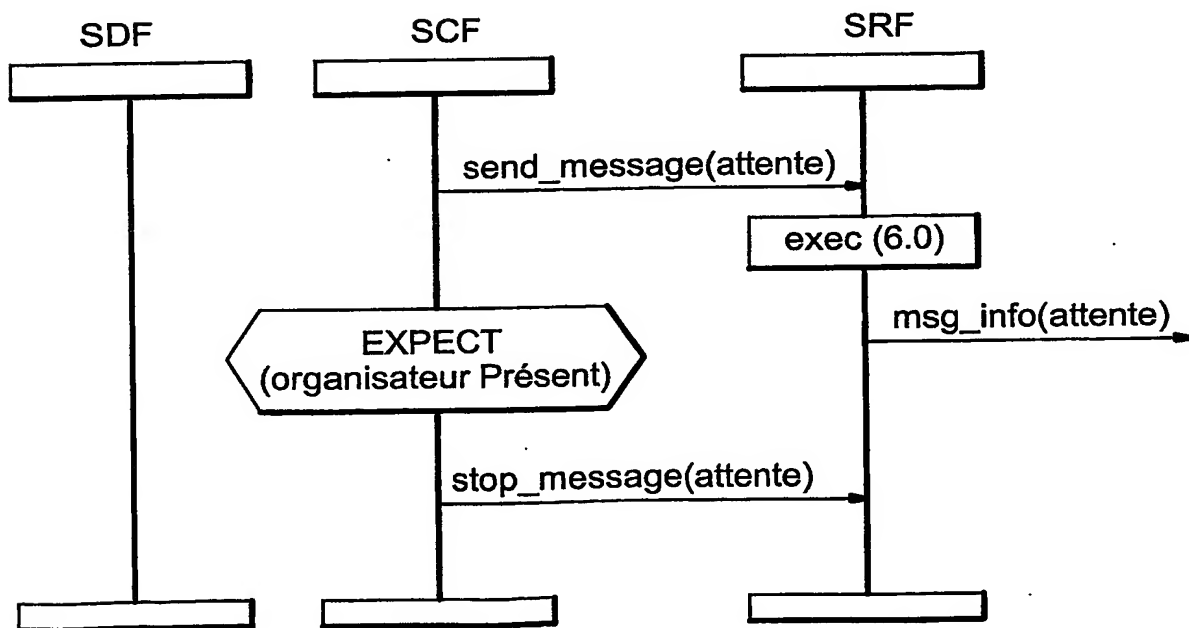


FIG. 19

Abandon Sur Attente

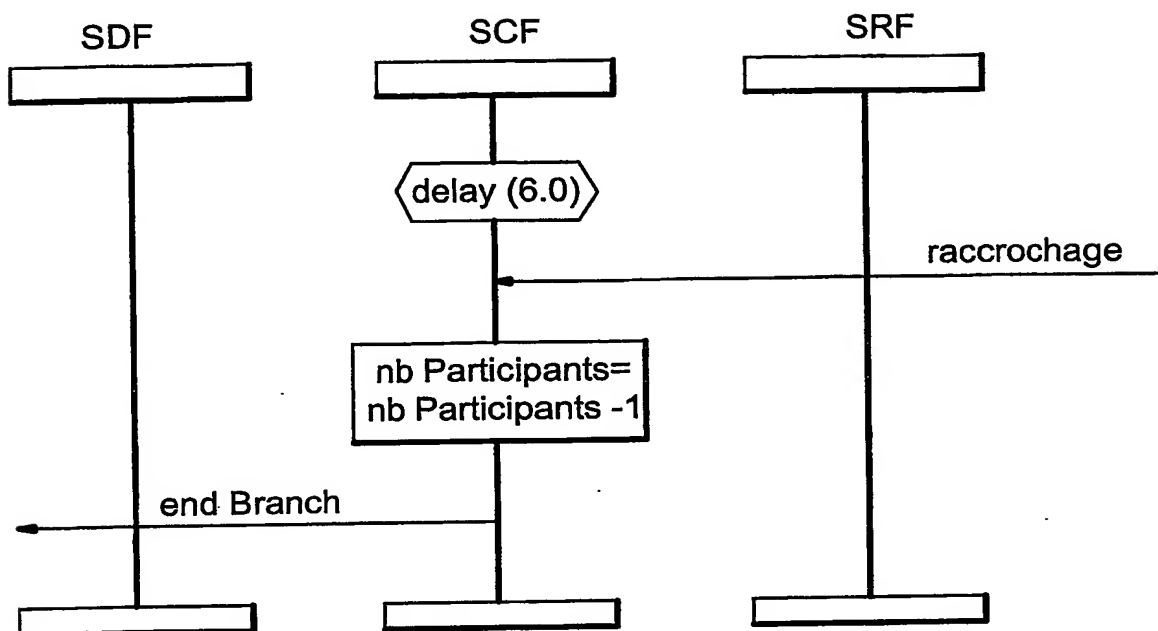


FIG. 20

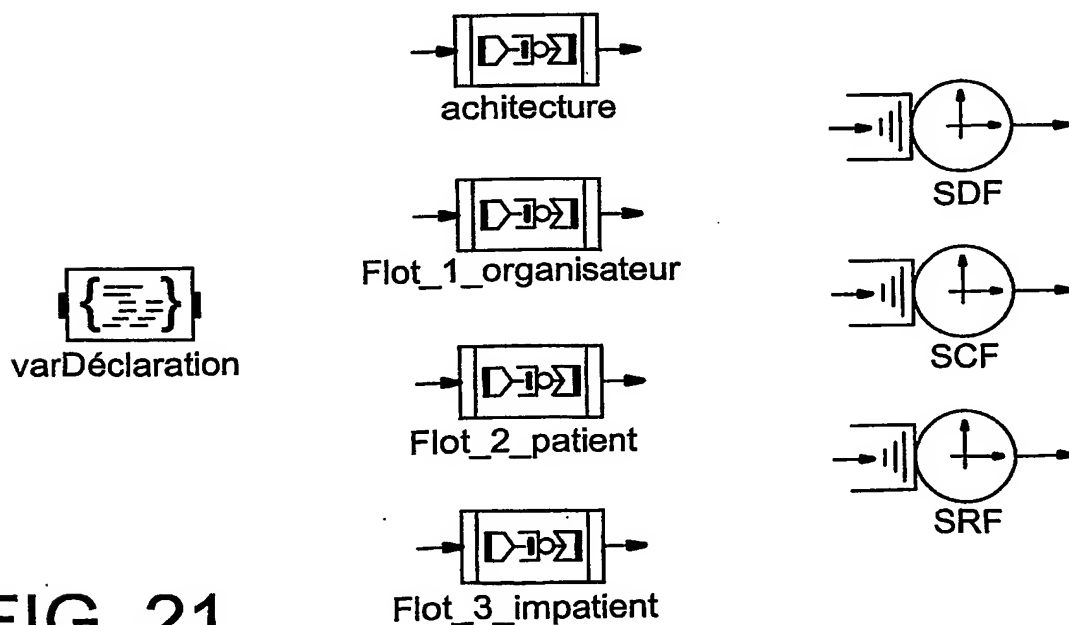


FIG. 21

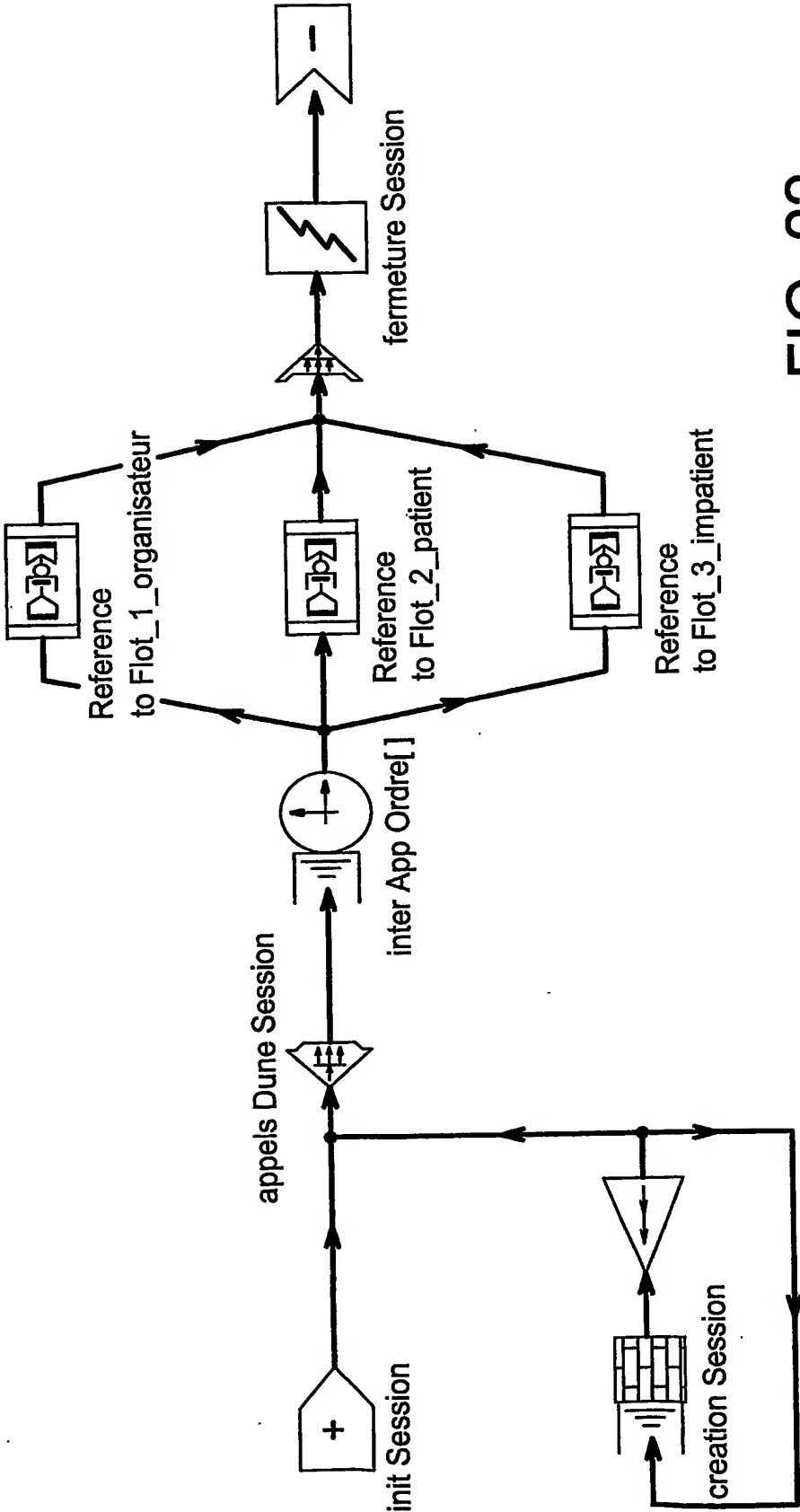


FIG. 22

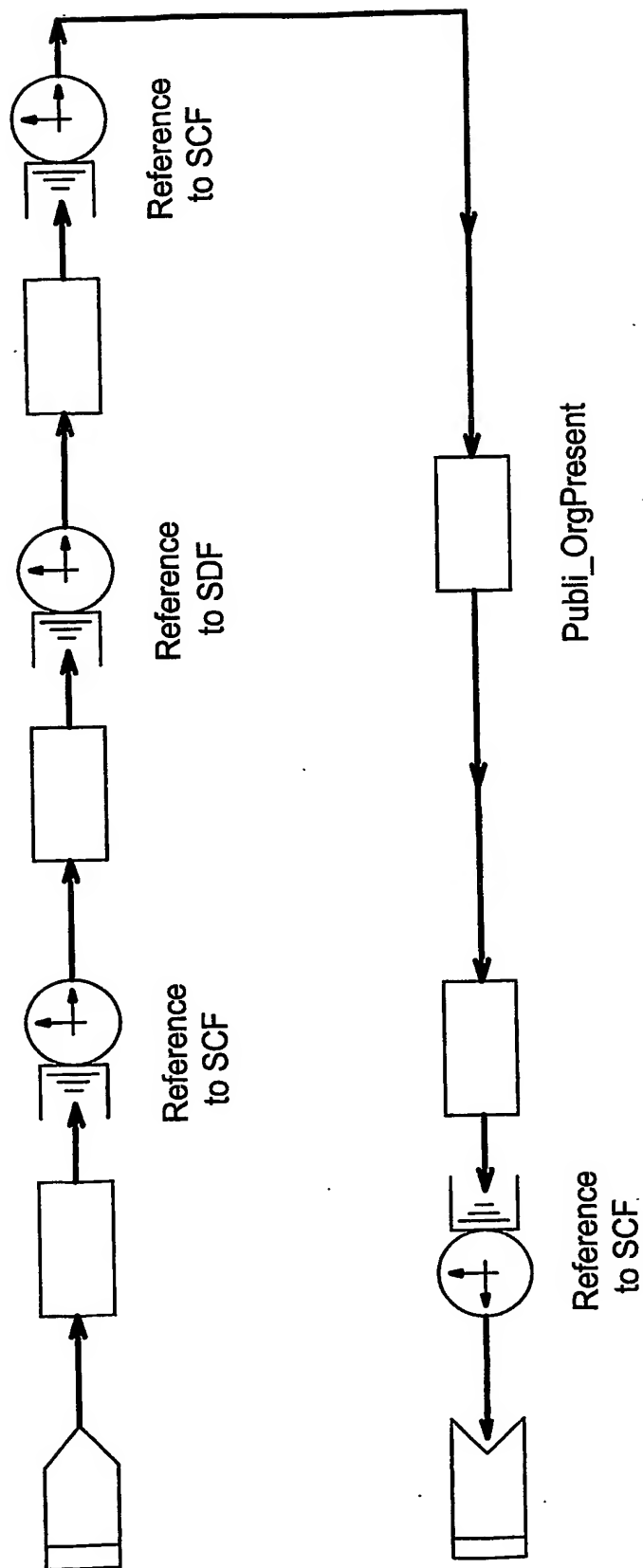


FIG. 23

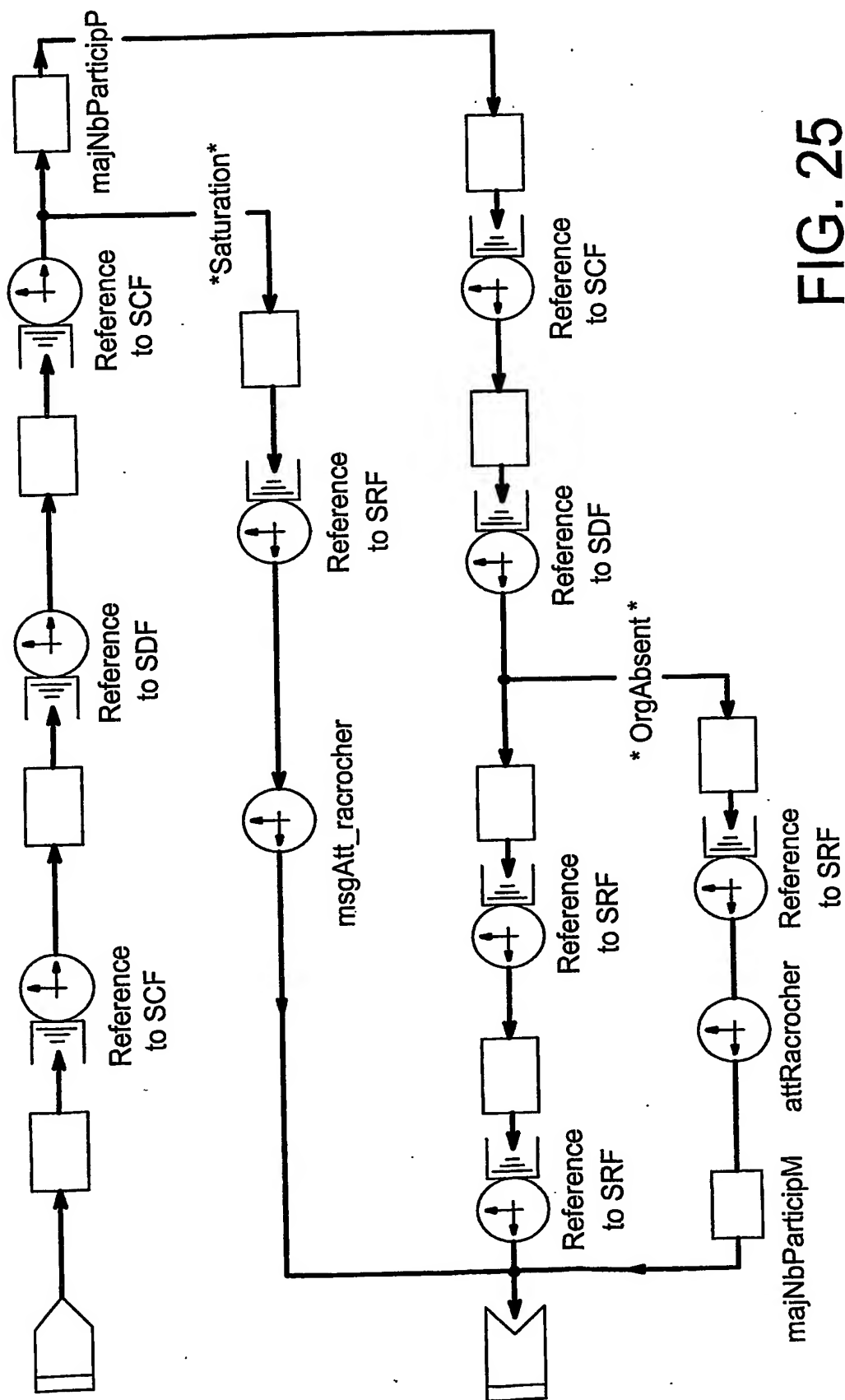


FIG. 25